

# 1. Artificial Intelligence and Knowledge Engineering in a Nutshell

This section provides supplemental information and details of artificial intelligence and knowledge engineering beyond what is provided by the section *Computational Thinking*<sup>1</sup> that is designed to be understood by professional accountants. If you do not want to dig into the details you can skip this section.

*(This is a combination of the old versions of knowledge engineering, problem solving logic, expert systems, and intelligent software agents. Much of this information has been moved into Computational Thinking. This section needs some heavy editing.)*

One type of practical knowledge is know-how<sup>2</sup>; how to accomplish something. Knowledge engineering is essentially the transformation of machine-readable instructions in such a way as to document how a system works or understanding how to make a system work the way you want that system to work. Brick-by-brick, much like building a house, knowledge engineers working with business domain experts and software engineers can create tools that automate certain types of tasks. Humans encode information, represent knowledge, and share meaning using machine-readable patterns, languages, and logic.

Professional accountants and auditors need to understand how computers work to properly understand the true capabilities of computers and the knowledge based systems that run on computers. They need to understand the capabilities of computers. They also need to understand how to control computers to get them to perform the work they desire to get them to perform that work performed in the manner they want the work performed. This understanding is necessary to effectively collaborate with information technology professionals and knowledge engineering professionals that build the tools professional accountants and auditors need and will be using in the Digital Age of accounting, reporting, auditing, and analysis<sup>3</sup>.

This understanding will become increasingly important as software is created to perform more and more tasks to assist professional accountants and auditors in their day-to-day work tasks. Machines augmenting humans to complete accounting, reporting, and auditing tasks will be the norm. Current advancements in areas such as artificial intelligence will contribute to an increased pace of change. This module summarizes, organizes, and synthesizes information helpful to professional accountants and auditors that want to gain this understanding.

## 1.1. Digital Environment and Machine-readable Information

Engineering is the application of a systematic<sup>4</sup>, disciplined<sup>5</sup>, quantifiable<sup>6</sup>, methodical<sup>7</sup>, rigorous<sup>8</sup> approach to the development, operation, and maintenance of

---

<sup>1</sup> Charles Hoffman, CPA, *Computational Thinking*, [http://www.xbrlsite.com/mastering/Part00\\_Chapter01.C\\_ComputerEmpathy.pdf](http://www.xbrlsite.com/mastering/Part00_Chapter01.C_ComputerEmpathy.pdf)

<sup>2</sup> Wikipedia, *Know-How*, <https://en.wikipedia.org/wiki/Know-how>

<sup>3</sup> *Getting Ready for the Digital Age of Accounting, Reporting and Auditing: a Guide for Professional Accountants*, <http://xbrlsite.azurewebsites.net/2017/Library/GettingReadyForTheDigitalAgeOfAccounting.pdf>

<sup>4</sup> Dictionary.com, *Systematic*, <http://www.dictionary.com/browse/systematic>

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.

Bridges are engineered when they are constructed. Engineering entails the skillful construction or creation of something leveraging known laws of how things interact with one another. A civil engineer does not simply throw concrete and steel together to construct a bridge. The bridge is engineered to balance cost, strength, likelihood that the bridge remains standing during high winds or an earthquake, etc. Building codes are created to help make sure good practices are used by engineers and builders.

Likewise when we work with information using a computer, how we achieve our goals and objectives is an engineering process, not simply throwing a few things together. How computers work is governed by laws that are well understood. What a computer can do reliably and safely are well understood by skillful computer science and information technology professionals. But there tends to be fewer “building codes” than might be appropriate for constructing software applications.

Professional accountants and auditors need is to understand how computers work and how to control the workings of computers to accurately understand what computers are capable of doing and what they are not capable of doing. Professional accountants need to understand how to get computers to do what they want them to do as these tools are increasingly important in today’s digital environment. Accountants have been called knowledge workers. The fact is everyone is a knowledge worker.

### **1.1.1. Understanding machine-readable knowledge**

Knowledge<sup>9</sup> is justified true belief. Knowledge is the fact or condition of being aware of something; the range of one’s information or understanding. Knowledge is justified with observable evidence that others can use to corroborate a belief, to support or justify the belief. Knowledge is provable.

In the past most knowledge was in human-readable form. For example, the knowledge of the financial condition and financial position of an economic entity was articulated in the form of a paper-based financial statement readable only by humans. In the age of paper, financial statements were marks on a surface. In a digital environment that same knowledge, through the use of structured data<sup>10</sup> formats such as XBRL, are machine readable bits of information organized in some sort of database. Structured data which represents knowledge in machine-readable form is being used more and more.

Does information, such as the financial position and financial condition of an economic entity, change based on the format used to represent that information? Clearly not. And so, to live in our digital world, professional accountants and auditors need to work effectively with machine-based information and knowledge

---

<sup>5</sup> Dictionary.com, *Disciplined*, <http://www.dictionary.com/browse/disciplined>

<sup>6</sup> Dictionary.com, *Quantifiable*, <http://www.dictionary.com/browse/quantifiable>

<sup>7</sup> Dictionary.com, *Methodical*, <http://www.dictionary.com/browse/methodical>

<sup>8</sup> Dictionary.com, *Rigorous*, <http://www.dictionary.com/browse/rigorous>

<sup>9</sup> Merriam-Webster, *Knowledge*, <http://www.merriam-webster.com/dictionary/knowledge>

<sup>10</sup> CFA Institute Calls for Broader and Deeper Use of Structured Data, <http://xbrl.squarespace.com/journal/2016/8/16/cfa-institute-calls-for-broader-and-deeper-use-of-structured.html>

represented as structured data used to convey meaning consistent with human-readable information.

*Digital* has positive features, but just like anything else it can also have potentially negative or less favorable features also. To harness the power of machines appropriately, professional accountants need to understand how these machines work and how to control them to get them to do what they want them to do.

### **1.1.2. Digital age is causing rapid change**

Whether you call it the information age, the digital age, or the era of cognitive computing<sup>11</sup>, change is occurring rapidly. Machines beat human chess masters. Machines play Jeopardy and win against human champions. The navigation systems in our cars perform amazing tasks that serve us well. Siri and other intelligent agents are at our beckon call.

Some tend to have an optimistic view of the capabilities of computers, overstating their potential. Others tend to have a pessimistic view of potential capabilities, understating possible usefulness. Understanding how technology works can help one be more conscious of the true capability of computers to help get work done.

With rapid change comes hype, snake oil salesmen looking for easy targets, expensive mistakes if the wrong choices are made, and missed opportunities if action is not taken. It is not necessary or even desirable to be on the bleeding edge of technology. But you don't want to completely miss the boat either.

*The Economist* predicts<sup>12</sup> that 94% of accounting jobs will be replaced by computers over the next 20 years. That percentage is 98% for accounting clerks, audit clerks, and bookkeepers. While predictions may, perhaps, be overstated; change to some degree is not only inevitable, that change is imminent.

Computers are machines. The first mechanical computers, called tabulating machines<sup>13</sup>, were created in the 1900s. Since then the effectiveness and efficiency of those machines have improved by orders of magnitude. But fundamentally the machines we use today are no different than mechanical tabulating machines of the past. The key word here is machine.

Increasingly; accounting, reporting, auditing, and analysis will be done in a digital environment.

### **1.1.3. Knowledge workers rearranging abstract symbols**

Computers sometimes seem to perform magic. But computers are really simply machines that follow very specific instructions. Skilled craftsmen, who wield their tools effectively, providing the correct machine-readable instructions, create what seems to be magic.

---

<sup>11</sup> *What is cognitive computing? IBM Watson as an example*, <http://www.duperrin.com/english/2014/05/27/whats-cognitive-computing-ibm-watson-example/>

<sup>12</sup> *The Economist*, Jan 18, 2014, *The future of jobs: The onrushing wave*, <http://www.economist.com/news/briefing/21594264-previous-technological-innovation-has-always-delivered-more-long-run-employment-not-less>

<sup>13</sup> Wikipedia, *Tabulating Machine*, [https://en.wikipedia.org/wiki/Tabulating\\_machine](https://en.wikipedia.org/wiki/Tabulating_machine)

In his book *Saving Capitalism*<sup>14</sup>, Robert Reich describes three categories that all modern work/jobs fit into:

- **Routine production services** which entails repetitive tasks,
- **In-person services** where you physically have to be there because human touch was essential to the tasks,
- **Symbolic-analytic services** which include problem solving, problem identification, and strategic thinking that go into the manipulation of symbols (data, words, oral and visual representations).

In describing the third category, symbolic-analytic services, Mr. Reich elaborates:

“In essence this work is to rearrange abstract symbols using a variety of analytic and creative tools - mathematical algorithms, legal arguments, financial gimmicks, scientific principles, powerful words and phrases, visual patterns, psychological insights, and other techniques for solving conceptual puzzles. Such manipulations improve efficiency-accomplishing tasks more accurately and quickly-or they better entertain, amuse, inform, or fascinate the human mind.”

Why this is interesting is the third category of work/jobs: symbolic-analytic services. Financial reporting, or at least many tasks related to financial reporting, fall into the symbolic-analytic service category.

How many professional accountants think of their job as "rearranging abstract symbols using a variety of analytic and creative tools?" Not many. Most professional accountants just do the work. Besides, what the heck is an "abstract symbol"?

Shelly Palmer breaks work tasks down in another way<sup>15</sup>. He points out that almost every human job requires us to perform some combination of the following four basic types of tasks:

- Manual repetitive (predictable)
- Manual nonrepetitive (not predictable)
- Cognitive repetitive (predictable)
- Cognitive nonrepetitive (not predictable)

**Manual** is using one's hands or physical action to perform work. **Cognitive** is using one's brain or mental action or a mental process of acquiring knowledge/understanding through thought, experience, use of the senses, or intuition. Predictable manual or cognitive tasks can be automated. Unpredictable manual or cognitive tasks cannot be automated. He gives the example of an assembly line worker that performs mostly manual repetitive tasks which, depending on complexity and a cost/benefit analysis, can be automated. On the other hand, a CEO of a major multinational conglomerate performs mostly cognitive nonrepetitive tasks which are much harder to automate.

---

<sup>14</sup> Robert B. Reich, *Saving Capitalism*, Alfred A. Knopf, page 204-206, (see a summary from *The Work of Nations* here [http://www.oss.net/dynamaster/file\\_archive/040320/e8eb8748abfe77204a145d5fbcc892fb/OSS1993-01-37.pdf#page=6](http://www.oss.net/dynamaster/file_archive/040320/e8eb8748abfe77204a145d5fbcc892fb/OSS1993-01-37.pdf#page=6))

<sup>15</sup> Shelly Palmer, *The 5 Jobs Robots Will Take Last*, <https://www.linkedin.com/pulse/5-jobs-robots-take-last-shelly-palmer>

#### 1.1.4. Computer empathy and computational thinking

*Psychology Today* defines empathy<sup>16</sup> as “the experience of understanding another person’s condition from their perspective”. Borrowing from that definition and modifying it slightly, think of **computer empathy** as the experience of understanding how computer software works from the perspective of the computer. Computer empathy is about understanding how a computer works so that you can better understand that tool and how to employ that tool in your craft to perform useful work reliably, repeatedly, predictably, and safely.

Computer empathy<sup>17</sup> is about demystifying accounting, reporting, auditing, and analysis in a digital environment. No magic; no metaphysics. Engineering.

This document provides a framework and principles to think about computers in a deliberate and conscious manner.

Another term for this is *computational thinking*<sup>18</sup>. **Computational thinking** is a thought process involved in formulating problems and their solutions so that the solutions are represented in a logical, clear, and systematic form that can be explained to and effectively carried out by a computer or human.

#### 1.1.5. Understanding what computers cannot do

Key to understanding what work computers are capable of performing is understanding and understanding of what computers are not capable of doing. Computers are good at repeating tasks over and over without variation. But computers are not good at any of the following sorts of tasks:

- Intuition
- Creativity
- Innovation
- Improvising
- Exploration
- Imagination
- Judgement (such as making a tough decision from incomplete information)
- Politics
- Law
- Unstructured problem solving
- Non-routine tasks
- Identifying and acquiring new relevant information
- Compassion

Some might argue that computers can be made to mimic some of the sorts of tasks in the list above. While such arguments might be valid, performance of computers in those sorts of tasks would likely be very costly and yield results that do not meet expectations. In other words, while theoretically possible using computers for such tasks, it is generally not practical.

---

<sup>16</sup> *Psychology Today*, *Empathy Basics*, <https://www.psychologytoday.com/basics/empathy>

<sup>17</sup> Charles Hoffman, CPA, *Computer Empathy*, <http://xbrlsite.azurewebsites.net/2018/Library/ComputerEmpathy.pdf>

<sup>18</sup> Center for Computational Thinking, Carnegie Mellon University, <https://www.cs.cmu.edu/~CompThink/>

### 1.1.6. Understanding information

Most business professionals understand the notion of data and may even understand a few things about relational databases. But data and information are not the same thing.

Relational databases store data. If you took a relational database out from under one software application and then connected it to a different software application would the second application understand the data in the database that was created to be used by the first application? The answer is no.

However, when you take an XBRL-based public company financial report out from an application (i.e. the creation software) and then connect it to another software application (i.e. the SEC Interactive Data Viewer) you can move the information and either application understands the exact same set of information. In fact, you could exchange that information to any of the 30 different software creation tools or other software vendors and each application would understand the information.

### 1.1.7. Difference between data, information, knowledge, and wisdom

There are specific differences between data, information, knowledge, and wisdom. This breakdown helps you understand the differences<sup>19</sup>:

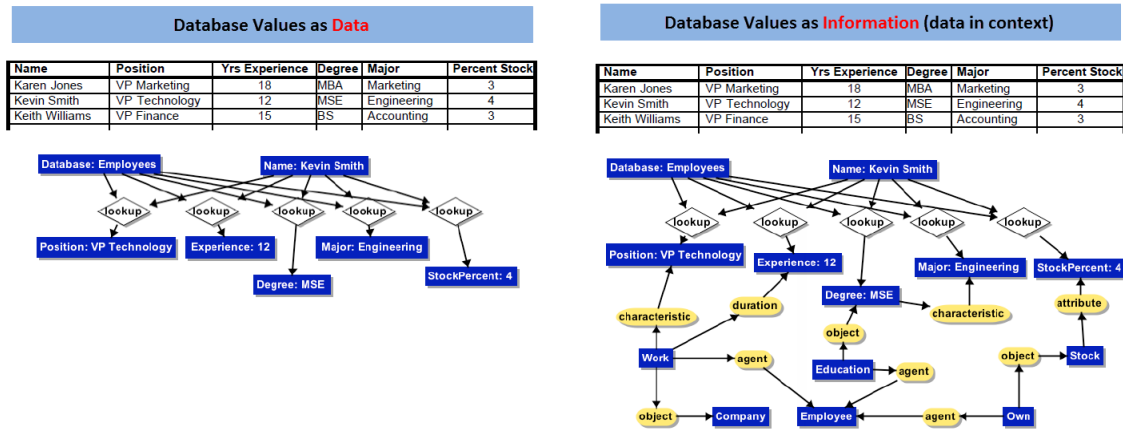
- **Data:** The basic compound for intelligence is data. Data are measures, observations, symbols, phenomenon, utterances, and other such representations of the world around us presented as external signals and picked up by various sensory instruments and organs. Simplified: data is raw facts and numbers.
- **Information:** Information is produced by assigning relevant meaning related to the context of the data to the data. Simplified: information is data in context.
- **Knowledge:** Knowledge is the understanding or interpretation, a justifiable true belief, of information and approach to act upon the information in the mind of the perceiver. Simplified: knowledge is the interpretation of information.
- **Wisdom (or Intelligence or Understanding):** Wisdom or intelligence wisdom embodies awareness, insight, moral judgments, and principles to construct new knowledge and improve upon existing understanding. Simplified: wisdom is the creation of new knowledge.

An absence of data is noise. This functional difference between data, information, knowledge, and wisdom is called the DIKW pyramid<sup>20</sup>.

---

<sup>19</sup> Gene Bellinger, Durval Castro, Anthony Mills; *Data, Information, Knowledge, and Wisdom*; Retrieved February 24, 2016, <http://www.systems-thinking.org/dikw/dikw.htm>

<sup>20</sup> Wikipedia, *DIKW Pyramid*, retrieved February 24, 2016; [https://en.wikipedia.org/wiki/DIKW\\_Pyramid](https://en.wikipedia.org/wiki/DIKW_Pyramid)



Information is data in context. That context information is generally not stored in a relational database. The graphic above shows the context information which is basically additional business rules that explain the data in more detail, put that data into context, turn the data into information, and then allow the information to be exchanged between different software systems.

### 1.1.8. Fundamental challenge: meaningful exchange of information

The fundamental challenge to get computers to perform useful work is the meaningful exchange of information between business systems. The only way a meaningful exchange of information can occur is the prior agreement as to<sup>21</sup>:

- technical syntax rules,
- business domain semantics rules, and
- business domain workflow rules.

ISO TR 9007:1987 (“Helsinki principles”) says<sup>22</sup> this in a slightly different way:

- Any meaningful exchange of utterances depends upon the prior existence of an agreed set of semantic and syntactic rules
- The recipients of the utterances must use only these rules to interpret the received utterances, if it is to mean the same as that which was meant by the utterer

### 1.1.9. Distinguishing technical syntax and domain semantics

One important aspect which you need to understand to understand the notion of a meaningful information exchange is the difference between syntax and semantics<sup>23</sup>.

- **Syntax** is *how* you say something
- **Semantics** is the *meaning* behind what you said

<sup>21</sup> Video, *Introduction to HL7*, slide 4, Retrieved February 24, 2016, [http://www.hl7.org/documentcenter/public\\_temp\\_D8292793-1C23-BA17-0C1CAB3A901C5581/training/IntroToHL7/player.html](http://www.hl7.org/documentcenter/public_temp_D8292793-1C23-BA17-0C1CAB3A901C5581/training/IntroToHL7/player.html)

<sup>22</sup> *Common Logic in Support of Metadata and Ontologies*, page 5, Retrieved June 24, 2016, [http://cl.tamu.edu/docs/cl/Berlin\\_OpenForum\\_Delugach.pdf](http://cl.tamu.edu/docs/cl/Berlin_OpenForum_Delugach.pdf)

<sup>23</sup> YouTube.com, *Introduction to the Semantic Web*, <https://www.youtube.com/watch?v=OGq8A2zfWKg>

Technical syntax is used to physically exchange information. Technical people are concerned with syntax. There are a handful of things that business professionals care about when it comes to technical syntax, mainly the power or expressiveness of the syntax.

Business professionals are far more concerned with semantics, the meaning behind what is being said. We will get into this in more detail later, for now just recognize that syntax and semantics are two different things.

#### **1.1.10. Understanding the important role of rules**

Rules prevent anarchy. The Merriam-Webster dictionary defines anarchy<sup>24</sup> as “a situation of confusion and wild behavior in which the people in a country, group, organization, etc., are not controlled by rules or laws.” Rules prevent information anarchy.

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.

Technical syntax rules arise from the best practices of information technology professionals. Business domain semantic rules arise from the best practices of knowledgeable business professionals. A business rule is a rule that 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 business 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 business rules is a choice of business professionals. The meaning of a business rule is separate from the level of enforcement someone might apply to the rule.

Please see the *Comprehensive Introduction to Business Rules for Professional Accountants*<sup>25</sup> for more information on the important topic of business rules.

#### **1.1.11. Understanding problem solving logic**

Computers work using the rules of mathematics. Mathematics works using the rules of logic. A problem solving logic is how a computer reasons.

To understand the notion of problem solving logic one first needs to understand the notion of logic and how logic can be applied to solving a problem.

A business rules engine processes business rules. The business rules processor/inference engine is the machine that processes the information. Some problem solving logic is used by the business rules processor. Every problem solving logic has some level of expressiveness. Problem solving logic is sometimes referred to as expressive power or reasoning capacity.

---

<sup>24</sup> Anarchy definition, Merriam-Webster, <http://www.merriam-webster.com/dictionary/anarchy>

<sup>25</sup> *Comprehensive Introduction to Business Rules for Professional Accountants*, <http://xbrlsite.azurewebsites.net/2016/Library/ComprehensiveIntroductionToBusinessRulesForProfessionalAccountants.pdf>



Please see the *Comprehensive Introduction to Problem Solving Logic*<sup>26</sup> for more information on the important topic of problem solving logic.

### 1.1.12. Understanding workflow rules

Workflow<sup>27</sup> the sequence of processes/tasks through which a piece of work passes from initiation to completion. There are two categories of business workflow systems or models that business rules should be able to express rules that operate in both worlds.

- **Process-centric workflows** generally use business rules at the workflow task level to manage workflow tasks.
- **Data-centric workflows** generally use business rules within workflows to make decisions about individual items of data.

This is not an "either/or" situation, but rather leveraging both workflow models in the design and execution of workflows is the way to go.

By combining the two different workflow models, business rules can be undertaken at both the task level for automating different decisions and at the data level for implementing filters over the data. Business rules can also be used to define operational features of a workflow, such as what to do when a specific task fails. Different standard approaches exist for representing workflow rules in machine-readable form including:

- Business Process Modeling (BPM)<sup>28</sup>
- XML Process Definition Language (XPDL)<sup>29</sup>
- Business Process Execution Language (BPEL)<sup>30</sup>
- Extensible Business Reporting Language (XBRL)

### 1.1.13. Shared view of reality to achieve a specific purpose

In his book<sup>31</sup> *Data and Reality*, William Kent provides an excellent summary that discusses the realities of sharing information. In Chapter 9: Philosophy in the Third Edition and Chapter 12: Philosophy in the first edition (which is available online) he paints a picture of why you want to go through the trouble of sharing information using machine-based processes and the realities of what that takes. This is what William Kent points out which I have paraphrased as it relates to financial reporting:

To create a shared reality to achieve a specific purpose: To arrive at a shared common enough view of "true and fair representation of financial information" such that most of our working purposes, so that reality does appear to be

---

<sup>26</sup> *Comprehensive Introduction to Problem Solving Logic*, [http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.5\\_ComprehensiveIntroductionToProblemSolvingLogic.pdf](http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.5_ComprehensiveIntroductionToProblemSolvingLogic.pdf)

<sup>27</sup> Wikipedia, *Workflow*, retrieved August 30, 2016, <https://en.wikipedia.org/wiki/Workflow>

<sup>28</sup> Wikipedia, *Business Process Modeling*, retrieved August 30, 2016, [https://en.wikipedia.org/wiki/Business\\_process\\_modeling](https://en.wikipedia.org/wiki/Business_process_modeling)

<sup>29</sup> Wikipedia, *XML Process Definition Language*, retrieved August 30, 2016, <https://en.wikipedia.org/wiki/XPDL>

<sup>30</sup> Wikipedia, *Business Process Execution Language*, retrieved August 30, 2016, [https://en.wikipedia.org/wiki/Business\\_Process\\_Execution\\_Language](https://en.wikipedia.org/wiki/Business_Process_Execution_Language)

<sup>31</sup> William Kent, *Data and Reality*, Technics Publications, (See this resource which has CHAPTER 12: Philosophy from the first version of this book, <http://www.bkent.net/Doc/darxrp.htm>)

objective and stable. So that you can query information reliably, predictably, repeatedly, safely.

Meaningful information exchange that is reliable, repeatable, predictable, safe, cost effective, easy to use, robust, scalable, secure when necessary, auditable (track provenance) when necessary.

Prudence dictates that using the information contained in a digital financial report should not be a guessing game. Safe, reliable, repeatable, predictable, reuse of reported financial information using automated machine-based processes is obviously preferable to a guessing game.

The effective meaningful exchange of information is created by skilled craftsmen that know their craft well. The craftsmen balances the system, bring the system into equilibrium to achieve a specific purpose. Creating this shared view of reality which allows this specific purpose to be achieved has benefit to the financial reporting supply chain.

That purpose should be clearly defined so that everyone understands the objective and exactly what the system can, and cannot, deliver.

## **1.2. Computers, Metadata, and Conceptual Models, Logic**

Computers are machines. Machine-readable metadata that is also understandable to humans is key in getting computers to perform work for business professionals. A conceptual model is metadata.

### **1.2.1. Machine-readable metadata which is also readable by humans**

Metadata<sup>32</sup> is simply data that provides information about other data. Machine-readable metadata adds perspective and context to data. People sometimes get into philosophical debates about what is data and what is metadata, but this is to completely miss the point.

This is what you need to know about metadata. Metadata is a good thing. More metadata is better. Standard metadata is even better. An example of metadata is the card catalog of a library. Metadata is generally organized into some sort of classification system.

There are three types of metadata<sup>33</sup>:

- **Descriptive:** describes and identifies information
- **Structural:** organizes the types and parts of information and how the parts are related to one another
- **Administrative:** provides other information that helps use some sort of system.

### **1.2.2. Three orders of order**

In his book *Everything is Miscellaneous*<sup>34</sup>, David Weinberger points out the three orders of order:

---

<sup>32</sup> Wikipedia, *Metadata*, <https://en.wikipedia.org/wiki/Metadata>

<sup>33</sup> YouTube, *Basics of Metadata*, <https://www.youtube.com/watch?v=-0vc6LeVa14>

- **First order of order.** Putting books on shelves is an example the first order of order. (data)
- **Second order of order.** Creating a list of books on the shelves you have is an example of second order of order. This can be done on paper or it can be done in a database. (metadata)
- **Third order of order.** Adding even more information to information is an example of third order of order. Using the book example; classifying books by genre, best sellers, featured books, bargain books, books which one of your friends has read; basically there are countless ways to organize something. (more metadata)

David Wenberger also points out that metadata has strategic implications. Third order removes the limitations which people seem to assume exist when it comes to organizing information. Wenberger says this about the third order of order:

“In fact, the third-order practices that make a company's existing assets more profitable, increase customer loyalty, and seriously reduce costs are the Trojan horse of the information age. As we all get used to them, third-order practices undermine some of our most deeply ingrained ways of thinking about the world and our knowledge of it.”

Thinking about tomorrow's systems and projecting yesterday's constraints onto those systems can limit creativity.

### 1.2.3. Classification systems

Things in the world are defined by their relations to one another; these explicit relations matter in creating logical definitions.

A classification system is a logical grouping of something based on some similarity or criteria. A classification system is a communications tool. A classification system structures information. A classification system can be informal or formal, more rigorously or less rigorously created, readable/usable by computers, or not. A classification system can be a controlled vocabulary. Classification systems can be classified as follows:

- A **dictionary** or list is a classification system that tends to provide descriptions without much, or any, structure. Dictionaries or lists simply provide a flat inventory of terms with no relations expressed between the terms. (But even a dictionary classifies terms into noun, verb, adverb, etc.)
- A **taxonomy** is a classification system which tends provide descriptions and a limited amount of structure generally in the form of one hierarchy into which some list of terms is categorized. Categories are basically sets. A taxonomy is a tree of categories of things with only one relation expressed so terms appear in only one location in a hierarchy of categories. A creator of a taxonomy creates concepts, creates coherent definitions for those concepts, and puts concepts into “buckets” or categories.
- An **ontology** is a classification system which tends to provide descriptions and multiple structures and therefore tends to have more than one hierarchy

---

<sup>34</sup> David Wenberger, *Everything is Miscellaneous*, Holt Paperbacks, 2007, page 17-23; <https://qoo.gl/oi8mkf>

into which terms are categorized. So an ontology can be thought of as a set of taxonomies. An ontology can express many different types of relations which includes traits/qualities of each term. An ontology is less like a tree and more like a graph<sup>35</sup> (network theory). This distinction is very important. The creator of an ontology identifies and establishes models explaining how things in a given ontology are related to one another, the kinds of relationships that exist, the rules of the model. If an ontology provides enough information, it can describe a conceptual model.

#### 1.2.4. Metacrap

In his essay *Metacrap: Putting the torch to seven straw-men of the meta-utopia*<sup>36</sup>, Cory Doctorow points out issues with metadata. When metadata is created these issues should be fore-front within one's mind so that you understand the strengths and weaknesses of your metadata.

#### 1.2.5. Conceptual model of the real world

A **model** is a set of entities and a set of relationships among those entities<sup>37</sup>. An **ontology**, if complete enough, can be a form of conceptual model (sometimes also called a logical model or entity-relationship model).

A **conceptual model**<sup>38</sup> is an abstraction of things that exist in the real world which is used to help people understand the subject or domain the model represents and build software applications. A conceptual model is composed of concepts, categories or type/classes of concepts, and rules which describe relations between types/classes of concepts.

A **theory**<sup>39</sup> is a prescriptive or normative statement which makes up a body of knowledge about what ought to be. A theory provides goals, norms, and standards. To theorize is to develop this body of knowledge.

A theory is a tool for understanding, explaining, and making predictions about a system. A theory describes absolutes. A theory describes the principles by which a system operates. A theory can be right or a theory can be wrong; but a theory has one intent: to discover the essence of some system.

A theory is consistent if its theorems will never contradict each other. Inconsistent theories cannot have any model, as the same statement cannot be true and false on the same system. But a consistent theory forms a conceptual model which one can use to understand or describe the system. A conceptual model or framework helps to make conceptual distinctions and organize ideas.

A *conceptual model*, *ontology*, and *theory* all tend to serve the same general purpose which is to describe a domain of knowledge.

Conceptual models, ontologies, and theories help us overcome the obstacles of getting a computer system to perform work. Conceptual models, ontologies, and

---

<sup>35</sup> Wikipedia, *Network Theory*, retrieved February 24, 2016; [https://en.wikipedia.org/wiki/Network\\_theory](https://en.wikipedia.org/wiki/Network_theory)

<sup>36</sup> Cory Doctorow, *Metacrap: Putting the torch to seven straw-men of the meta-utopia*, <https://people.well.com/user/doctorow/metacrap.htm>

<sup>37</sup> John F. Sowa, *Fads and Fallacies about Logic*, page 2, <http://www.ifsowa.com/pubs/fflogic.pdf>

<sup>38</sup> Wikipedia, *Conceptual Model*, retrieved August 14, 2016; [https://en.wikipedia.org/wiki/Conceptual\\_model](https://en.wikipedia.org/wiki/Conceptual_model)

<sup>39</sup> Wikipedia, *Theory*, retrieved August 29, 2016, <https://en.wikipedia.org/wiki/Theory>

theories are formal specifications. Formal specifications are precise, concise and unambiguous. Formal specifications are communications tools. If these are machine-readable and therefore machine-checkable notation, a wide variety of automated checks can be applied to test the conceptual model, ontology, or theory to see if they act as designed. The disciplined approach of using formal specifications means that subtle errors and oversights will be detected and corrected.

And so conceptual models, ontologies, and theories which are machine-readable serve two roles: first, to describe and second to verify against that description. When creating information it is important to verify that what has been created is consistent with the expected description. When consuming information it is important to understand that the information being consumed is consistent with the expected description. Remember: nonsense-in-nonsense-out.

### 1.2.6. Logical theory

So, a **theory** is a formal statement of rules about some subject that describes and otherwise explains the nature of that subject. A theory describes some aspect of the world and tries to describe the principles by which that aspect of the world operates. A theory can be right or wrong, but a theory is characterized by its intent: the discovery of essence. A theory does not simplify. A theory describes absolutes. A successful theory can become a fact. A theory is a tool for contemplating something with an intent to gain insight or understanding.

**Logic** a set of principles that form a framework for correct reasoning. Logic is a process of deducing information correctly. Logic is about the correct methods that can be used to prove a statement is true or false. Logic tells us exactly what is meant. Logic allows systems to be proven.

In logic, a **statement** is a sentence that is either true or false. You can think of statements as pieces of information that are either correct or incorrect. And therefore, statements are pieces of information that you apply logic to in order to derive other pieces of information which are also statements. Statements are basically rules.

A **logical theory** is a set of logical statements that formally describes some subject or system. **Axioms**<sup>40</sup> are statements that describe self-evident logical principles that no one would argue with. **Theorems**<sup>41</sup> are logical deductions which can be proven by constructing a chain of reasoning by applying axioms and the rules of logic in the form of IF...THEN statements.

A **rule**, or business rule or assertion, is a true statement with respect to some model of the real world that could possibly exist given some logical theory. You cannot create rules that are true in worlds that can never exist. A rule can be a mathematical expression. A rule is a type of logical statement.

### 1.2.7. Need for a framework

A conceptual model, ontology, or logical theory serves as a framework. For example, an XBRL-based digital financial report has a framework. The following are definitions which help you understand what a framework provides:

- A framework is a broad overview or overarching conceptual structure.

---

<sup>40</sup> Wikipedia, *Axiom*, <https://en.wikipedia.org/wiki/Axiom>

<sup>41</sup> Wikipedia, *Theorem*, <https://en.wikipedia.org/wiki/Theorem>

- A framework is a system or set of principles, assumptions, ideas, concepts, values, rules, laws, agreements, and practices that constitutes a way of viewing reality or establishes the way something operates.

In a Ted Talk about brain science, Jeff Hawkins points out the need for a framework and theory<sup>42</sup>. Things look complicated until you understand them. Once you understand them you can create a framework and theory.

What is conspicuously lacking from the XBRL International, XBRL US, the FASB and the SEC is a broad framework let alone a theory on how to think about digital financial reports. And that is why the *Financial Report Semantics and Dynamics Theory*<sup>43</sup> was created.

### 1.2.8. Systems thinking

A **system**<sup>44</sup> is a regularly interacting or interdependent group of items forming a unified whole.

Dr. W. Edwards Deming explains systems in the video *A Theory of Systems for Educators and Managers*<sup>45</sup>. Deming explains that the typical way of managing a complex system is to take the system, break it into parts, and then try and manage each part as well as possible. But that does not work because it is possible to improve the performance of each part, and destroy the system as a whole. Deming put it this way when describing systemic thinking:

“Working together is the main contribution to systemic thinking as opposed to working apart separately.”

There is a difference between analysis and synthesis:

- **Analysis** is separate the whole into parts and study each part individually. Analysis is the dominant mode of thought in the western world. You cannot explain the behavior of a system by analysis. You can reveal its structure and see how it works, but you cannot understand why it works the way it works.
- **Synthesis** the combination of ideas to form a theory or system. If you want to understand why something works the way it does you use synthesis to figure that out.

You need both analysis and synthesis. Analysis tells you how. Synthesis tells you why. If you want to find out how something works you analyze it. If you want to understand why it works the way it does, you use synthesis. You cannot explain the behavior of a system through analysis.

Working together is the primary benefit of systemic thinking. This is as opposed to working apart separately. The performance of the whole is not the sum of the performance of the parts separately. The performance of a system is the product of the interactions of the parts of the system.

---

<sup>42</sup> Ted Talk, *Jeff Hawkins: How brain science will change computing*, <https://www.youtube.com/watch?v=G6CVj5IQkzk>

<sup>43</sup> *Financial Report Semantics and Dynamics Theory*, <http://xbrl.squarespace.com/fin-report-sem-dyn-theory/>

<sup>44</sup> Wikipedia, *System*, <https://en.wikipedia.org/wiki/System>

<sup>45</sup> YouTube.com, *A Theory of Systems for Educators and Managers*, <https://www.youtube.com/watch?v=2MJ3IGJ4OFo>

Idealized redesign is thinking creatively about a system. Assume a system was completely destroyed and you could do whatever you want right now to replace the system. If you don't know what to do when you could do if you can do whatever you want; how could you possibly know what to do if you can't do whatever you want?

### 1.2.9. Formal systems

A formal system<sup>46</sup> is defined as any well-defined system of abstract thought based on the model of mathematics. Basically, formal systems can be explained and proven to work or show system flaws and inconsistencies using the language of mathematics. Every formal system has some sort of formal language<sup>47</sup> that explains that system. Every formal system can be tested to see if it works using a formal proof<sup>48</sup>.

### 1.2.10. Formal logic

A **logic** can be defined as any precise notation for expressing statements that can be judged to be either true or false<sup>49</sup>.

Aristotle<sup>50</sup> is said to be the father of formal logic. Logic is a discipline of philosophy<sup>51</sup>. Logic<sup>52</sup> is the study of correct reasoning. Logic is the science of argument evaluation. You evaluate arguments using the rules of logic to see if the argument holds to be true. An argument is a set of statements<sup>53</sup>.

The notation of what we call elementary school arithmetic took centuries to develop<sup>54</sup>. But today we take mathematics for granted.

Formal logic is the basis for mathematics. Mathematics is a formal system. Formal logic is the basis for describing theories and proving theories.

Formal logic was consciously broken into two groups: **first-order logic**<sup>55</sup> and **higher-order logic**<sup>56</sup>. There is a reason for this. Systems based on first-order logic can be proven to be **sound** (all provable theory statements are true in all models) and **complete** (all theory statements which are true in all models are provable using proof theory).

Higher-order logics are less well-behaved than those of first-order logic. They are less predictable and therefore less reliable and they are significantly harder to implement using computers. That is why computer systems are generally based on first-order logic.

This is all well understood by good software engineers.

---

<sup>46</sup> Wikipedia, *Formal System*, retrieved August 29, 2016, [https://en.wikipedia.org/wiki/Formal\\_system](https://en.wikipedia.org/wiki/Formal_system)

<sup>47</sup> Wikipedia, *Formal Language*, retrieved August 29, 2016, [https://en.wikipedia.org/wiki/Formal\\_language](https://en.wikipedia.org/wiki/Formal_language)

<sup>48</sup> Wikipedia, *Formal Proof*, retrieved August 29, 2016, [https://en.wikipedia.org/wiki/Formal\\_proof](https://en.wikipedia.org/wiki/Formal_proof)

<sup>49</sup> John F. Sowa, *Fads and Fallacies about Logic*, page 2, <http://www.ifsowa.com/pubs/fflogic.pdf>

<sup>50</sup> Wikipedia, *Aristotle*, retrieved August 29, 2016, <https://en.wikipedia.org/wiki/Aristotle>

<sup>51</sup> Wikipedia, *Philosophy*, retrieved August 29, 2016, <https://en.wikipedia.org/wiki/Philosophy>

<sup>52</sup> Wikipedia, *Logic*, retrieved August 29, 2016, <https://en.wikipedia.org/wiki/Logic>

<sup>53</sup> *Crash Course in Formal Logic, Part 1*, <https://www.youtube.com/watch?v=ywKZgjpMBUU>

<sup>54</sup> *Set Theory and Foundations of Mathematics*, retrieved August 29, 2016, <http://settheory.net/>

<sup>55</sup> Wikipedia, *First-order Logic*, retrieved August 29, 2016, [https://en.wikipedia.org/wiki/First-order\\_logic](https://en.wikipedia.org/wiki/First-order_logic)

<sup>56</sup> Wikipedia, *Higher-order Logic*, retrieved August 29, 2016, [https://en.wikipedia.org/wiki/Higher-order\\_logic](https://en.wikipedia.org/wiki/Higher-order_logic)

### 1.2.11. Need to limit first-order logic

The full set of first-order logic is not decidable so it must be limited. Description logics<sup>57</sup> are a family of representational languages. *SROIQ* Description Logic<sup>58</sup> is one such language which is based on a fragment of first-order logic that is decidable. However, *SROIQ* Description Logic does not include the ability to represent mathematical computations because the complete set of mathematics is not decidable.

Another form of first-order logic is PROLOG which is a programming language<sup>59</sup>. PROLOG a general purpose logic programming language that is also declarative. PROLOG is based on first-order logic. The syntax of PROLOG is derived from Horn clauses<sup>60</sup> which is a subset of first-order logic that is decidable. Because PROLOG is declarative, program logic is expressed represented by facts, relations, and rules. Questions are asked and then answers are provided based on the facts, relations, and rules.

PROLOG has some undesirable aspects and so it was modified even further resulting in DATALOG<sup>61</sup>. DATALOG is both sound and complete.

What is the exact set of first-order logic which should be used to represent systems so that they are both sound and complete and maximize the expressive power of the language? The answer to this question is found in the next section.

### 1.2.12. Understanding the importance of boundaries

First-order logic is very powerful and can be used to express a theory which fully and categorically describes structures of a finite domain (problem domain). This is achieved by specifying the things of the problem domain and the relations between those things.

No first-order theory has the strength to describe an infinite domain. Essentially what this means is that the things and the relations between things which make up a problem domain must have distinct boundaries. They must be made finite.

This is not to say that such a system cannot be flexible. For example, a form is not flexible. A financial report is not a form. This is not to say, however, that a financial report cannot be finite.

Extensibility is the ability to add things to a system. *Local extensibility* is extensibility that is "inside the walls" of one organization and all extensibility is explicitly coordinated and controlled within and by one organization. For example, a chart of accounts of an organization is an example of local extensibility. You have a framework for adding accounts and you can add whatever accounts you need to the systems.

*Distributed extensibility* is extensibility that is not explicitly controlled and coordinated by one specific organization but rather using standards-based mechanisms and rules. For example, XBRL-based financial reports submitted to the

---

<sup>57</sup> Description Logics, see [http://en.wikipedia.org/wiki/Description\\_logic](http://en.wikipedia.org/wiki/Description_logic)

<sup>58</sup> *A Description Logic Primer* describes the importance of *SROIQ*, <http://arxiv.org/pdf/1201.4089.pdf>

<sup>59</sup> Understanding the Importance of PROLOG to Digital Financial Reporting, <http://xbrl.squarespace.com/journal/2015/7/23/understanding-the-importance-of-prolog-to-digital-financial.html>

<sup>60</sup> Wikipedia, *Horn Clause*, retrieved August 29, 2016, [https://en.wikipedia.org/wiki/Horn\\_clause](https://en.wikipedia.org/wiki/Horn_clause)

<sup>61</sup> Wikipedia, *DATALOG*, retrieved August 29, 2016, <https://en.wikipedia.org/wiki/Datalog>



U.S. SEC is a type of distributed extensibility because while the entire system is controlled by some standard set of rules, each reporting entity has control and can extend the system; but they must stay within a set of rules which coordinates the extensibility.

The point is that one must correctly understand the notion of finite and boundaries. Even a distributed system which is extensible can have solid boundaries if the system is engineered correctly.

### 1.2.13. Describing systems formally using a logic framework

Deliberate, rigorous, conscious, skillful execution is preferable to haphazard, negligent, unconscious, inept execution if you want to be sure something works. Engineering a system to make sure it works as designed is a very good thing.

A digital financial report<sup>62</sup> is a type of formal system. A digital financial report is mechanical and those mechanical aspects of how such a report works can be described using a logical theory and conceptual model. The *Financial Report Semantics and Dynamics Theory*<sup>63</sup> describes the conceptual model of a digital financial report.

A system such as the digital financial report needs to be described precisely so that professional accountants understand the mechanics of how the system works so that the system can be used effectively and so the system works how the system was intended to work.

Logic or more precisely business logic can be used to describe a system. There are various types of logic frameworks.

**Z Notation**<sup>64</sup> is an ISO/IEC standard for describing systems precisely. Z Notation is used to describe safety-critical systems such as nuclear power plants, railway signaling systems, and medical devices. But Z Notation is not machine-readable.

**Common Logic**<sup>65</sup> (CL), also an ISO/IEC standard, is a framework for a family of logic languages, based on first-order logic, intended to facilitate the exchange and transmission of knowledge in computer-based systems. Common Logic is machine-readable. Further, the logic allowed to be expressed by Common Logic is consciously limited to avoid logical catastrophes<sup>66</sup> which cause systems to break.

Common Logic is about being practical, something business professionals generally tend to like. Common Logic is a conscious compromise in order to achieve reliability, predictability, and safety. Common Logic is a "sweet spot" that achieves high expressivity but consciously gives up certain specific things that lead to catastrophic results that cause systems to potentially break making a system unsound; so that a system will be sound. Common Logic establishes boundaries, allowing creators of

---

<sup>62</sup> *Conceptual Overview of an XBRL-based, Structured Digital Financial Report*, <http://xbrl.azurewebsites.net/2016/Library/ConceptualOverviewOfDigitalFinancialReporting.pdf>

<sup>63</sup> *Financial Report Semantics and Dynamics Theory*, <http://xbrl.squarespace.com/fin-report-sem-dyn-theory/>

<sup>64</sup> *Understanding the Importance of Z Notation*, <http://xbrl.squarespace.com/journal/2015/9/4/understanding-the-importance-of-z-notation.html>

<sup>65</sup> *Understanding Common Logic*, <http://xbrl.squarespace.com/journal/2016/6/23/understanding-common-logic.html>

<sup>66</sup> *Brainstorming the Idea of Logical Catastrophes or Failure Points*, <http://xbrl.squarespace.com/journal/2015/7/25/brainstorming-idea-of-logical-catastrophes-or-failure-points.html>

systems to "stay within the lines" and if you do, you get a maximum amount of expressiveness with the minimum risk of catastrophic system failure. Thus, you get a more reliable, dependable system.

**Semantics of Business Vocabulary and Business Rules**<sup>67</sup> (SBVR) is an OMG standard that was designed and built to be logically equivalent to Common Logic.

What is the point? Ask yourself why ISO/IEC and OMG would go through the trouble to create specifications such as Z Notation, Common Logic, and Semantics of Business Vocabulary and Business Rules? The answer to that question is to enable systems to be described precisely so that they can be implemented successfully using computer software but also enables interoperability between different systems.

Logics can be used to describe systems. Standard logics, such as Common Logic and Semantics of Business Vocabulary and Business Rules enable interoperability. As John F. Sowa put it in *Fads and Fallacies about Logic*<sup>68</sup>:

"In summary, logic can be used with commercial systems by people who have no formal training in logic. The fads and fallacies that block such use are the disdain by logicians for readable notations, the fear of logic by nonlogicians, and the lack of any coherent policy for integrating all development tools. The logic-based languages of the Semantic Web are useful, but they are not integrated with the SQL language of relational databases, the UML diagrams for software design and development, or the legacy systems that will not disappear for many decades to come. A better integration is possible with tools based on logic at the core, diagrams and controlled natural languages at the human interfaces, and compiler technology for mapping logic to both new and legacy software."

The bottom line is that the best balance between expressive power and safe implementation has been achieved by the ISO/IEC global standard Common Logic. **Common Logic**<sup>69</sup> is a framework for a family of logic languages, based on first-order logic, intended to facilitate the exchange and transmission of knowledge in computer-based systems. That safely expressive sweet spot is also used by the OMG standard **Semantics of Business Vocabulary and Business Rules**<sup>70</sup> which was consciously designed to be logically equivalent to ISO/IEC Common Logic.

**Rulelog**<sup>71</sup> is a logic that is consciously engineered to be consistent with ISO/IEC Common Logic and OMG Semantics of Business Vocabulary and Business Rules. Rulelog is a dialect of W3C's RIF<sup>72</sup>. RuleML<sup>73</sup> is a syntax for implementing rules. Other standard and proprietary syntaxes exist for implementing rules.

**SHACL**<sup>74</sup> (Shapes Constraint Language) is a logic that is consciously engineered to work in a "closed world" similar to a relational database. SHACL is a W3C

---

<sup>67</sup> OMG, *Semantics of Business Vocabulary and Business Rules (SBVR)*, section 2.5 Conformance of an SBVR Processor, page 7, <http://www.omg.org/spec/SBVR/1.0/>

<sup>68</sup> John F. Sowa, *Fads and Fallacies about Logic*, page 6, <http://www.jfsowa.com/pubs/fflogic.pdf>

<sup>69</sup> *Understanding Common Logic*, <http://xbrl.squarespace.com/journal/2016/6/23/understanding-common-logic.html>

<sup>70</sup> OMG, *Semantics of Business Vocabulary and Business Rules (SBVR)*, section 2.5 Conformance of an SBVR Processor, page 7, <http://www.omg.org/spec/SBVR/1.0/>

<sup>71</sup> *Rulelog*, <http://ruleml.org/rif/rulelog/spec/Rulelog.html>

<sup>72</sup> W3C, *RIF Overview (Second Addition)*, <http://www.w3.org/TR/rif-overview/>

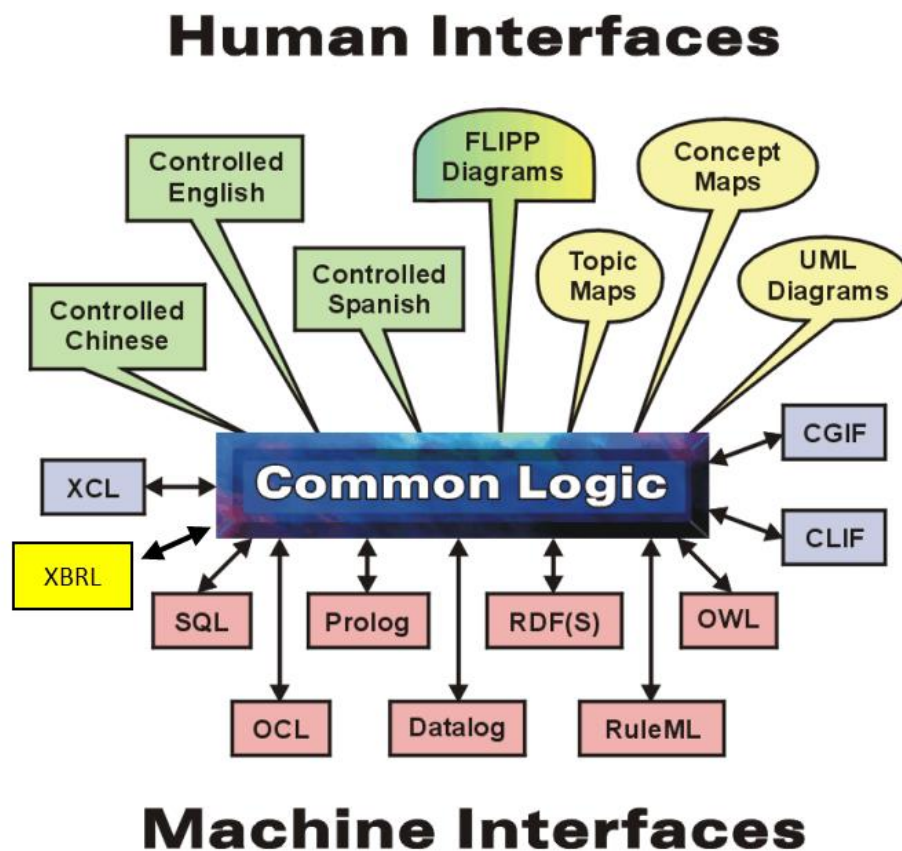
<sup>73</sup> *RuleML*, [http://wiki.ruleml.org/index.php/RuleML\\_Home](http://wiki.ruleml.org/index.php/RuleML_Home)

<sup>74</sup> W3C, *Shapes Constraints Language*, <http://www.w3.org/TR/shacl/>

recommendation. SHACL is a language for validating RDF graphs against a set of conditions. SHACL is used to perform closed-world constraint checks on RDF-based data.

The most important thing to realize is that there is a good, safe target in terms of an expressive logic that is also safely implementable in software so catastrophic failures are avoided. Another very good thing is that business professionals don't need to understand the underlying technical details of these logic standards, nor will they every have to deal with them. Higher level languages that follow the foundations set by Common Logic, Semantics of Business Vocabulary and Business Rules, Rulelog, and SHACL. XBRL, if implemented correctly, can achieve this objective.

The following graphic shows the role Common Logic<sup>75</sup> plays, establishing a family of logical dialects shared between different software syntax implementations: (note that this graphic was modified, XBRL was added)



The language in which a problem is stated has no effect on complexity. Reducing the expressive power of a logic does not solve any problems faster; its only effect is to make some problems impossible to state<sup>76</sup>.

<sup>75</sup> John F. Sowa, *Common Logic: A Framework for a Family Of Logic-Based Languages*, page 5, <http://www.ifsowa.com/ikl/SowaST08.pdf>

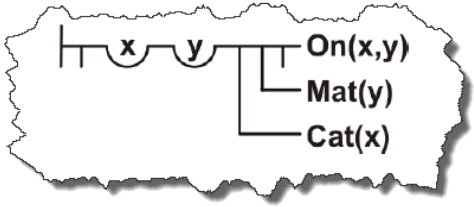
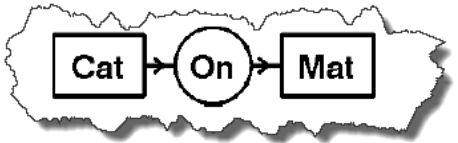
<sup>76</sup> John F. Sowa, *Fads and Fallacies about Logic*, page 5, <http://www.ifsowa.com/pubs/fflogic.pdf>

**1.2.14. Multiple technology stacks**

People have different preferences. The article *Whatever Happened to the Semantic Web*<sup>77</sup> points out issues related to creating a general semantic web as contrast to a specific semantic web. Software engineers found RDF and the W3C’s semantic web stack too complicated. Software engineers preferred JSON to XML because they found XML too hard to work with. Fads, trends, preferences, and other issues point to the fact that there will always be multiple technology stacks used to solve technical problems and issues.

**1.2.15. How to say “A cat is on a mat.”**

John Sowa provides a simple example that makes a profound point. How to you represent the notion that “a cat is on a mat” in various technical syntax forms<sup>78</sup>. Here are the examples he provides:

	$\Sigma_x \Sigma_y \text{Cat}_x \cdot \text{Mat}_y \cdot \text{On}_{x,y}$
$\exists x \exists y \text{Cat}(x) \wedge \text{Mat}(y) \wedge \text{On}(x,y)$	<pre>SELECT FIRST.ID, SECOND.ID FROM OBJECTS FIRST, OBJECTS SECOND, SUPPORTS WHERE FIRST.TYPE = "Cat" AND SECOND.TYPE = "Mat" AND SUPPORTS.SUPPORTER = SECOND.ID AND SUPPORTS.SUPPORTEE = FIRST.ID</pre>
$(\text{exists } ((x \text{ Cat}) (y \text{ Mat})) (\text{On } x \ y))$	$[\text{Cat } *x] [\text{Mat } *y] (\text{On } ?x \ ?y)$
	<p>A cat is on a mat.</p>

Which of these syntax do you find the easiest to read? Which syntax conveys the most meaning to you? Which of these technical syntax would be easiest for a business professional to work with? How hard is it to convert one syntax to another syntax? One approach is to use logic symbols<sup>79</sup> to represent words. Another approach is to use natural language to explain logic.

<sup>77</sup> Whatever Happened to the Semantic Web, <https://twobithistory.org/2018/05/27/semantic-web.html>

<sup>78</sup> John F. Sowa, *Common Logic: A Framework for a Family Of Logic-Based Languages*, page 2-3, <http://www.ifsowa.com/ikl/SowaST08.pdf>

<sup>79</sup> Wikipedia, *Logic Symbols*, [https://en.wikipedia.org/wiki/List\\_of\\_logic\\_symbols](https://en.wikipedia.org/wiki/List_of_logic_symbols)

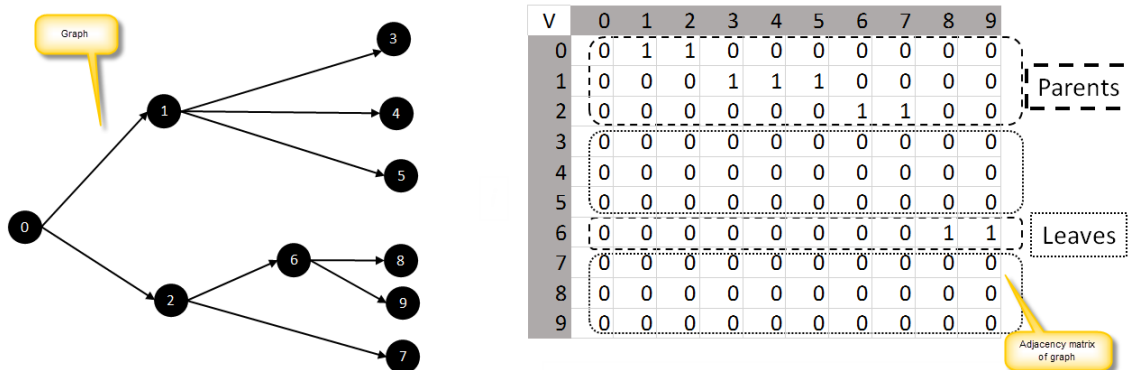
### 1.2.16. Understanding difference between a tree and a graph

Graph theory<sup>80</sup> is a useful communications tool. A graph is an abstract idea that does not really exist in the real world. But, graph theory can be used to describe and explain real world information and structural relationships that can be reduced to a graph. The graph is not the real world; the graph is just used to explain how the real world works. Network theory<sup>81</sup> is the study of graphs. Network theory is a part of graph theory. A network can be defined as a graph in which nodes and/or edges have attributes such as a “name” or a “role”. A graph is defined in mathematical terms by the structural information contained in its adjacency matrix<sup>82</sup>. The elements of the adjacency matrix indicate whether pairs of vertices are adjacent or not in the graph.

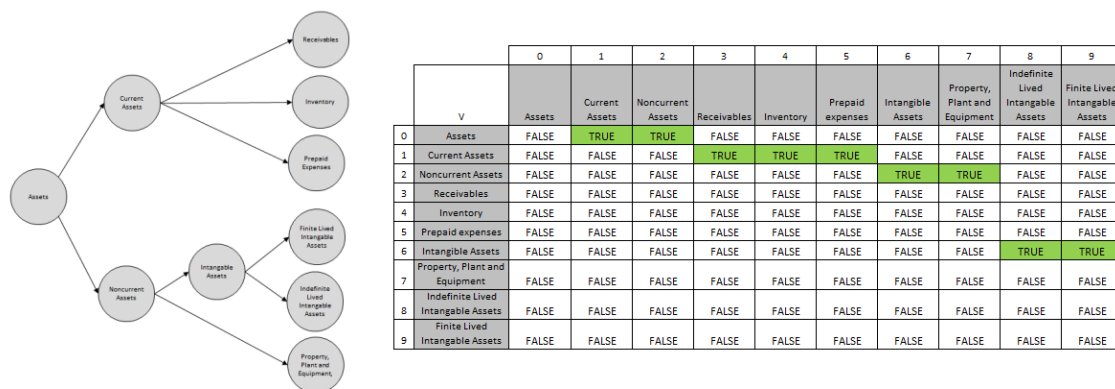
Basically, graph theory is useful in helping professional accountants represent their world and explain that world to computers in their terms which is mathematics.

Below are two diagrams. The first diagram is an abstract graph and the adjacency matrix for the graph. The second diagram is a more concrete graph of accounting terms. The two diagrams provide the same meaning.

Abstract graph and adjacency matrix:



More concrete graph of accounting relations and adjacency matrix:



<sup>80</sup> Wikipedia, *Graph Theory*, [https://en.wikipedia.org/wiki/Graph\\_theory](https://en.wikipedia.org/wiki/Graph_theory)

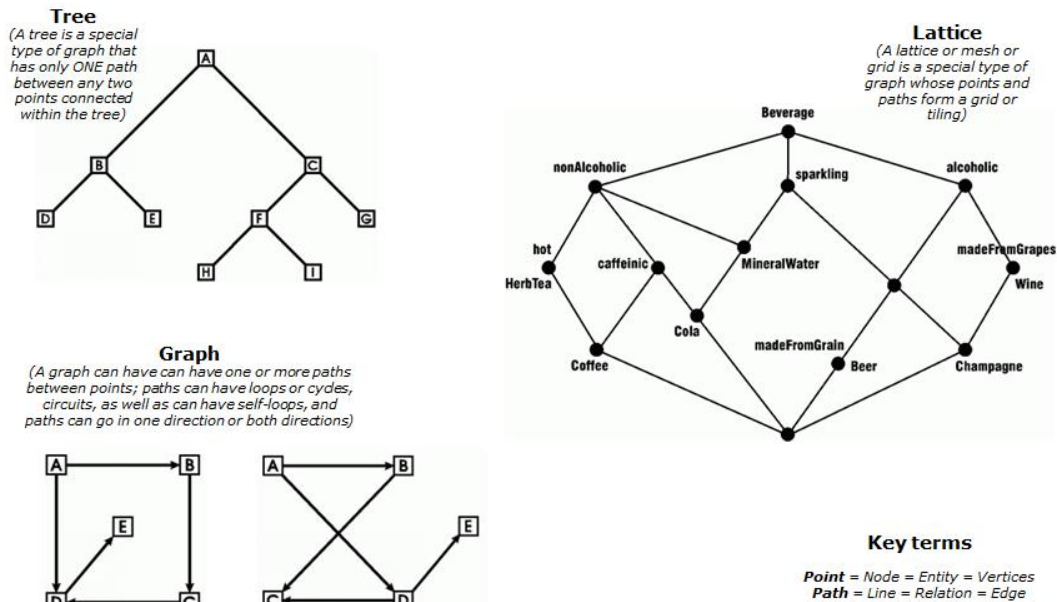
<sup>81</sup> Wikipedia, *Network Theory*, [https://en.wikipedia.org/wiki/Network\\_theory](https://en.wikipedia.org/wiki/Network_theory)

<sup>82</sup> Wikipedia, *Adjacency Matrix*, [https://en.wikipedia.org/wiki/Adjacency\\_matrix](https://en.wikipedia.org/wiki/Adjacency_matrix)

One of the big problems accountants have related to XBRL is that they don't understand the difference between a "tree" or hierarchy and a "graph". Why is this important? Graph theory is a way of thinking, reasoning, and solving problems. If you look at the Seven Bridges of Königsberg<sup>83</sup> problem you can get an appreciation for the utility of graph theory.

The following graphics provide a good overview of the difference between a tree and a graph. These key terms help you understand the diagrams<sup>84</sup>:

- Point = Node = Entity = Vertices
- Path = Line = Relation = Edge



- **Graph:** A graph can have one or more paths between points; paths can have loops or cycles, circuits, as well as can have self-loops, and paths can go in one direction or both directions.
- **Tree:** A tree is a special type of graph that has only one path between any two points connected within the tree.
- **Lattice:** A lattice or mesh or grid is a special type of graph whose points and paths form a grid or tiling.

Graphs are used to explain the relations between objects. Even this basic information is helpful when discussing relations between information. Do you want cycles to exist? Are their multiple relations?

Directed acyclic graphs<sup>85</sup> are the "sweet spot" where you get all the advantages of trees, some of the advantages of graphs, but without the catastrophic consequences

<sup>83</sup> Wikipedia, Seven Bridges of Königsberg , [https://en.wikipedia.org/wiki/Seven\\_Bridges\\_of\\_K%C3%B6nigsberg](https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg)

<sup>84</sup> John F. Sowa, *Mathematical Background, Graphs*, <http://www.ifsowa.com/logic/math.htm#Graph>

that can be caused by certain types of cycles. For example, spreadsheets and tables can be modeled as a directed acyclic graph.

### 1.2.17. *Unique name assumption*

Some conceptual modelling tools, such as OWL, assume non-unique names whereas in real life most people expect named things to be distinct. This can become problematic when trying to represent the real world using a conceptual model. An example helps you understand this issue<sup>86</sup>,

“Two sons and two fathers went to a pizza restaurant. They ordered three pizzas. When they came, everyone had a whole pizza. How can that be?”

Most people make the mistake of assuming that two sons and two fathers meant that there were four people. Of course there were only three people: a grandfather, a father and a son.

Naming things can be tricky<sup>87</sup>. Remember, computers are dumb beasts. Mechanisms must exist to help you be conscious of what you are doing and provide feedback such as a line of reasoning and transparent explanation mechanisms to help the user of a system make certain that the system is operating as intended. This news list post succinctly summarizes the problems of the unique name assumption (UNA)<sup>88</sup>.

There are plenty of things in the universe that are known by many names. That's why the phrase "a.k.a." exists. That's why owl:sameAs exists. UNA has advantages for the consumers of data, but it has huge drawbacks for publishers. The Web, and therefore the Semantic Web, is based on the principle that you don't have to coordinate to publish things. You never have to agree on a specific identifier to say something about the world.

In relational databases, however, things are more coordinated. The point is not that UNA is right or wrong; the point is that one must be conscious of whether UNA is assumed or not<sup>89</sup>.

### 1.2.18. *Understanding why logical catastrophes break systems*

A logical catastrophe is a failure point. Logical catastrophes must be eliminated. Systems should never have these failure points. A basic example of a catastrophic failure is creating metadata that puts a process into an infinite loop that the software will not recover from. This type of catastrophic failure is resolved by simply not allowing the conceptual model to include such structures which cause the possibility of infinite loops. It really is that straight forward.

In network theory<sup>90</sup> there is a relation type called a directed cycle<sup>91</sup> which can cause infinite loops. The following graphics of a directed and undirected cycle helps you understand the potential problems of directed cycles and infinite loops:

---

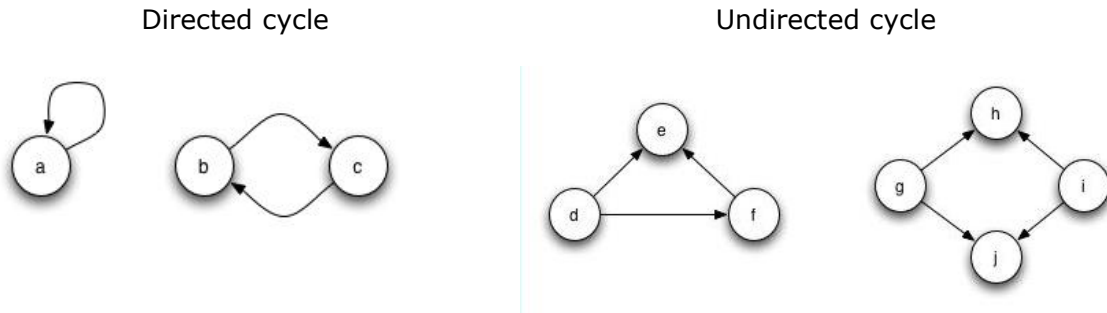
<sup>85</sup> Wikipedia, *Directed acyclic graphs*, retrieved May 17, 2018, [https://en.wikipedia.org/wiki/Directed\\_acyclic\\_graph](https://en.wikipedia.org/wiki/Directed_acyclic_graph)

<sup>86</sup> Ian Davis, *Unique Name Assumption*, <http://blog.iandavis.com/2005/04/unique-name-assumption/>

<sup>87</sup> Patrick McKenzie, *Falsehoods Programmers Believe About Names*, <http://www.kalzumeus.com/2010/06/17/falsehoods-programmers-believe-about-names/>

<sup>88</sup> Unique name assumption in OWL, <https://stackoverflow.com/questions/14164344/unique-name-assumption-in-owl>

<sup>89</sup> W3C, *SHACL Use Cases and Requirements*, <https://www.w3.org/TR/shacl-ucr/#uc16:-constraints-and-controlled-reasoning>



Directed cycles should be avoided and don't generally exist in many areas of reality. Tools for representing reality should not allow their users to unintentionally create directed cycles. Business professionals should be conscious of the difference between directed and undirected cycles.

Here are other types of logical catastrophes:

- **Undecidability:** If a question cannot be resolved to a TRUE or FALSE answer; for example if the computer returns UNKNOWN then unpredictable results can be returned. Logic used by a computer must be decidable.
- **Infinite loops:** If a computer somehow enters an infinite loop from which it cannot return because of a logic error or because the logic is too complex for the machine to work with; the machine will simply stop working or return nonsense.
- **Unbounded system structures or pieces:** Systems need boundaries for them to work correctly. If a system does not have the proper boundaries, then a machine can become confused or not understand how to work with information that is provided. For example, if an entirely new class of concept is added to a system that the system has no knowledge of, the system will not understand how to process that class of concept and will fail.
- **Unspecific or imprecise logic:** Confusing precise results with the capabilities of a computer to provide a statistically created result can cause problems. It is not expected that the business system at the level of describing the things in the system be able to support "fuzzy logic" or "probabilistic reasoning" or other such functionality.

The article *The Semantic Web and the Business Rules Approach ~ Differences and Consequences*<sup>92</sup> by Silvie Spreeuwenberg provides this list of issues which should be considered when implementing a logic:

- closed world vs. open world assumption,
- higher order logic vs. first order logic,
- horn clause logic vs. predicate logic,
- deontic logic vs. non-modal languages,

<sup>90</sup> Network Theory, [https://en.wikipedia.org/wiki/Network\\_theory](https://en.wikipedia.org/wiki/Network_theory)

<sup>91</sup> Directed cycle described by the XBRL Technical Specification, <http://www.xbrl.org/Specification/XBRL-2.1/REC-2003-12-31/XBRL-2.1-REC-2003-12-31+corrected-errata-2013-02-20.html#Directed-cycles>

<sup>92</sup> Silvie Spreeuwenberg, *The Semantic Web and the Business Rules Approach ~ Differences and Consequences*, <https://www.brcommunity.com/articles.php?id=b257>



- based purely on formal model theory vs. based on axioms,
- negation as failure vs. scoped negation as failure.

Business professionals do not need to understand the details of these logical catastrophes. What business people need to understand is whether the technical people they work with understand these logical catastrophes and how to avoid them.

### **1.2.19. Understanding the critical importance of decidability**

There are two fundamental approaches to viewing a system that one could take: the *open world assumption* and the *closed world assumption*. Formal logic and relational databases use the closed world assumption. Decidability means that a conclusion can be reached.

- In the **open world assumption** a logical statement cannot be assumed true on the basis of a failure to prove the logical statement. On a World Wide Web scale this is a useful assumption; however a consequence of this is that an inability to reach a conclusion (i.e. not decidable).
- In the **closed world assumption** the opposite stance is taken: a logical statement is true when its negation cannot be proven; a consequence of this is that it is always decidable. In other applications this is the most appropriate approach.

So each type of system can choose to make the open world assumption or the closed world assumption based on its needs. Because it is important that a conclusion as to the correct mechanics of a financial report is required because consistent and correct mechanics are necessary to making effective use of the information contained within a financial report; the system used to process a financial report must make the closed world assumption.

### **1.2.20. Setting the right expectation by understanding the capabilities of computers**

First-order logic has limitations<sup>93</sup>. Business professionals need to understand these limitations so that they understand what computers can and cannot do, what is hard and what is easy to implement using computers, and to otherwise set their expectations appropriately. Remember, computers cannot perform magic. Computers fundamentally follow the rules of mathematics which follow the rules of formal logic. It really is that straight forward.

Reasoning with uncertainty can be important if knowledge is incomplete. Techniques exist for dealing with uncertainty such as confidence levels. Business domain professionals need to understand when such reasoning is being used.

It is difficult to get computers to effectively work with information such as the following:

- fuzzy expressions: "It **often** rains in autumn."
- non-monotonicity: "Birds fly, penguin is a bird, but penguin does not fly."
- propositional attitudes: "Eve **thinks** that 2 is not a prime number." (It is true that she thinks it, but what she thinks is not true.)

---

<sup>93</sup> Martin Kuba, Institute of Computer Science, *OWL 2 and SWRL Tutorial, Limitations of First-order logic expressiveness*, <http://dior.ics.muni.cz/~makub/owl/>

- modal logic<sup>94</sup>
  - possibility and necessity: “It is **possible** that it will rain today.”
  - epistemic modalities: “Eve **knows** that 2 is a prime number.”
  - temporal logic: “I am **always** hungry.”
  - deontic logic<sup>95</sup>: “You **must** do this.”

While it is possible to implement this sort of functionality within computer systems using technologies such as probabilistic reasoning<sup>96</sup>, those systems will be less reliable and significantly more difficult to create.

### 1.2.21. *Limitations of classification systems*

David Wenberger's book *Everything is Miscellaneous* points out two important things to understand about classification systems:

- Every classification scheme ever devised inherently reflects the biases of those that constructed the classification system.
- The role metadata plays in allowing you to create your own custom classification system so you can have the view of something that you want.

Metadata and the correct architecture provide the flexibility necessary to create the sort of classification system you might desire which could be different that the desires of the creators of a classification system.

### 1.2.22. *Need for a thick metadata layer, knowledge acquisition*

The key ingredient in an expert system is domain knowledge. The power of any expert system is proportional to the high-quality domain knowledge available. What is not in dispute is the need for a "thick metadata layer" and the benefits of that metadata in terms of getting a computer to be able to perform useful and meaningful work. But what is sometimes disputed, it seems, is *how* to most effectively and efficiently get that thick metadata layer. There are two basic approaches to getting this metadata layer:

- **Have the computer figure out what the metadata is:** This approach uses artificial intelligence, machine learning, and other high-tech approaches to detecting patterns and figuring out the metadata.
- **Tell the computer what the metadata is:** This approach leverages business domain experts and knowledge engineers to piece together the metadata so that the metadata becomes available.

Because knowledge acquisition can be slow and tedious, much of the future of expert systems depends on breaking the knowledge acquisition bottleneck and in codifying and representing a large knowledge infrastructure. However, this is not an “either/or” question. Both manual and automated knowledge acquisition methods can be used together.

---

<sup>94</sup> Stanford Encyclopedia of Philosophy, *Modal Logic*, *What is Modal Logic?*, <https://plato.stanford.edu/entries/logic-modal/#WhaModLog>

<sup>95</sup> Stanford Encyclopedia of Philosophy, *Deontic Logic*, <https://plato.stanford.edu/entries/logic-deontic/>

<sup>96</sup> Wikipedia, *Probabilistic Logic*, retrieved August 28, 2016, [https://en.wikipedia.org/wiki/Probabilistic\\_logic](https://en.wikipedia.org/wiki/Probabilistic_logic)

There is a lot of talk about neural networks<sup>97</sup> enabling things like machine learning<sup>98</sup> and deep learning<sup>99</sup>. There are two important points that business professionals tend to miss or software vendors creating such software tend to leave out of their sales pitches. First, the amount of training<sup>100</sup> that is necessary to get a neural network to work correctly. The training process is time consuming, expensive, and error prone.

Second, because the process is error prone; there are good uses for neural networks figuring out the “thick layer of metadata”, and there are very bad uses. One description of what neural networks are best for is the following:

Neural networks are universal approximators, and they work best if the system you are using them to model has a high tolerance to error. One would therefore not be advised to use a neural network to balance one's cheque book! However they work very well for: <sup>101</sup>

- capturing associations or discovering regularities within a set of patterns;
- where the volume, number of variables or diversity of the data is very great;
- the relationships between variables are vaguely understood; or,
- the relationships are difficult to describe adequately with conventional approaches.

And so, the probability of a neural network figuring out something like the US GAAP Financial Reporting XBRL Taxonomy is basically zero. However, that said; if such metadata is created and then that human-created metadata is used to train neural networks the probability that the neural network can create something useful goes up dramatically.

So, again, this is not an “either-or” proposition. This is about using the right tool for the right job and not being misguided by snake oil salesmen who don't have your interest in mind.

### **1.2.23. Comparing expressiveness**

Expressiveness is the set of things that can possibly be expressed by some language or logic. Below is a graphic which shows the relative expressiveness of Common Logic and Z Notation relative to the universe of all possible expressiveness<sup>102</sup>.

---

<sup>97</sup> A Basic Introduction To Neural Networks, <http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html>

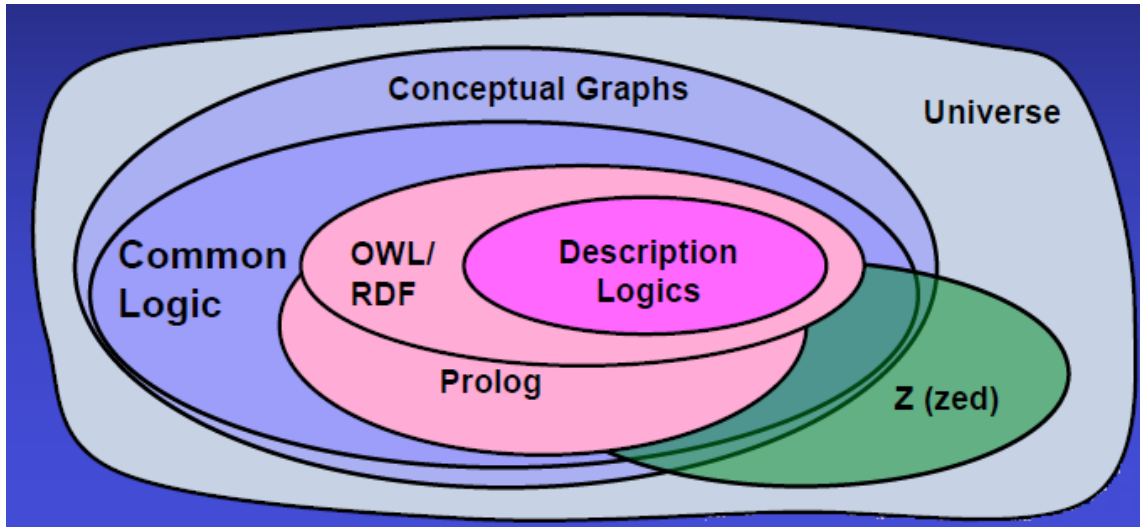
<sup>98</sup> Wikipedia, *Machine Learning*, retrieved August 29, 2016, [https://en.wikipedia.org/wiki/Machine\\_learning](https://en.wikipedia.org/wiki/Machine_learning)

<sup>99</sup> Wikipedia, *Deep Learning*, retrieved August 29, 2016, [https://en.wikipedia.org/wiki/Deep\\_learning](https://en.wikipedia.org/wiki/Deep_learning)

<sup>100</sup> Hugo Larochelle, et. al., *Deep Learning and Applications in Neural Networks*, <http://www.slideshare.net/hammawan/deep-neural-networks>

<sup>101</sup> A Basic Introduction To Neural Networks, *What Applications Should Neural Networks Be Used For?*, <http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html>

<sup>102</sup> *Common Logic in Support of Metadata and Ontologies*, Page 2, Retrieved June 24, 2016, [http://cl.tamu.edu/docs/cl/Berlin\\_OpenForum\\_Delugach.pdf](http://cl.tamu.edu/docs/cl/Berlin_OpenForum_Delugach.pdf)



Not even included in this comparison, because the expressiveness is so low is the Comma Separated Values<sup>103</sup> (CSV) technical format. CSV is a very popular data format and it is very easy to use. But CSV does nothing to help assure the quality of data represented in this technical format. Basically, you can articulate a simple list in CSV and you cannot provide information which helps a user of the information understand that the information is consistent with expectations in terms of representation (i.e. you cannot figure out the quality of the list).

#### **1.2.24. Understanding the relation between expressiveness and reasoning capacity**

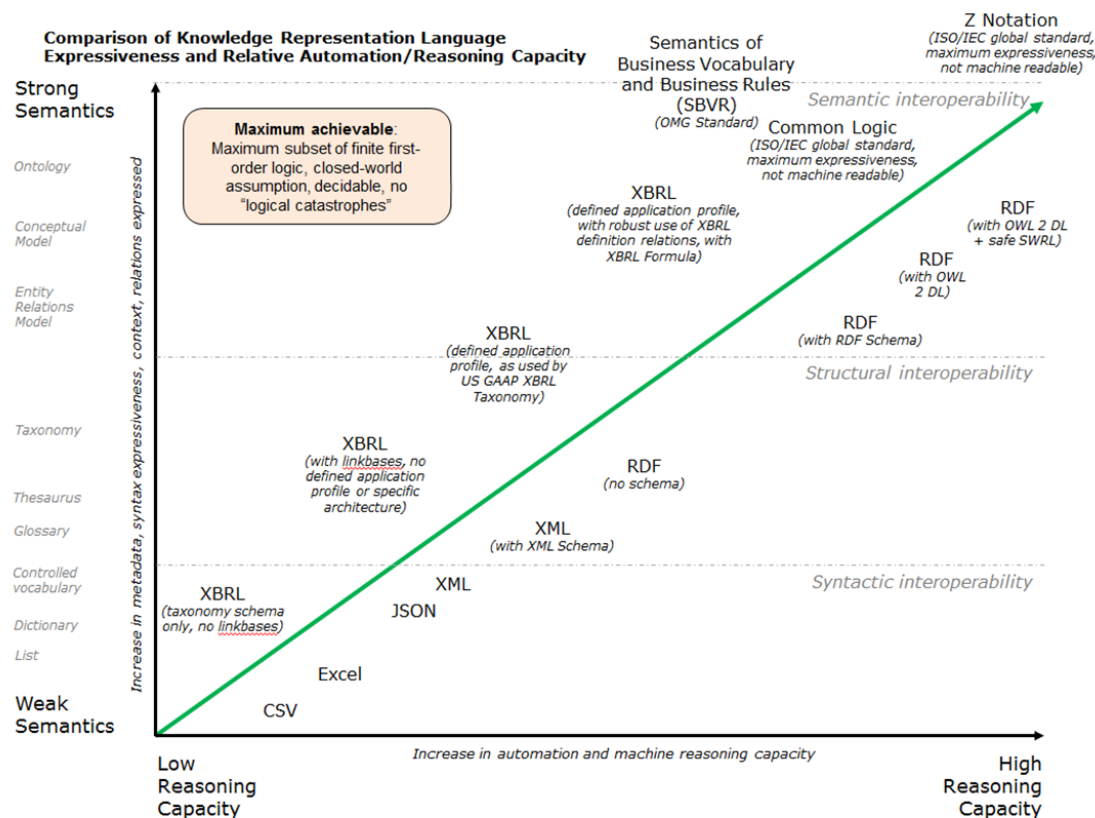
Why is the expressiveness of a language important? There are two reasons. First, the more expressive a language the more that language can provide in terms of describing the information being represented and verifying the consistency of what is being represented with expectations (i.e. quality).

But secondly, the more expressive the language is; the more a computer can do for a user of an application in terms of reasoning capacity. So, the two work together. Both the quality of the information being processed is higher and what the software can do is higher because of both the expressiveness of the language but also because of the quality of the information which is represented.

Another way to say this is “nonsense in, nonsense out”. As has been pointed out, the only way to have a meaningful exchange of information is the prior existence of technical syntax rules (the language syntax), business domain semantics (the descriptive and structural metadata), and the workflow rules (protocols for what to do if say an amended financial report is submitted to a regulator).

This graphic below compares the relative knowledge representation language expressiveness and the relative automation and reasoning capacity which is achievable using that language.

<sup>103</sup> Wikipedia, *Comma Separated Values*, retrieved August 28, 2016, [https://en.wikipedia.org/wiki/Comma-separated\\_values](https://en.wikipedia.org/wiki/Comma-separated_values)



Inspired by similar comparisons from *An Intrepid Guide to Ontologies* <http://www.mkbergman.com/date/2007/05/16/> and *Semantics Overview* <http://prezi.com/prvxi8po3in/semantics-overview/>

At the bottom left hand corner of the graphic you see “CSV” which is not expressive (i.e. weak semantics). At the top left you see the ISO/IEC standard “Z Notation” which is highly expressive (i.e. strong semantics). But remember, Z Notation is not machine-readable. But you also see Common Logic, Semantics of Business Vocabulary, and XBRL as having strong semantics. Those three formats are all machine-readable.

No knowledge representation language is 100% complete. Each has specific, knowable limitations. One must be conscious of such limitations when creating a representation of some problem domain in machine readable form.

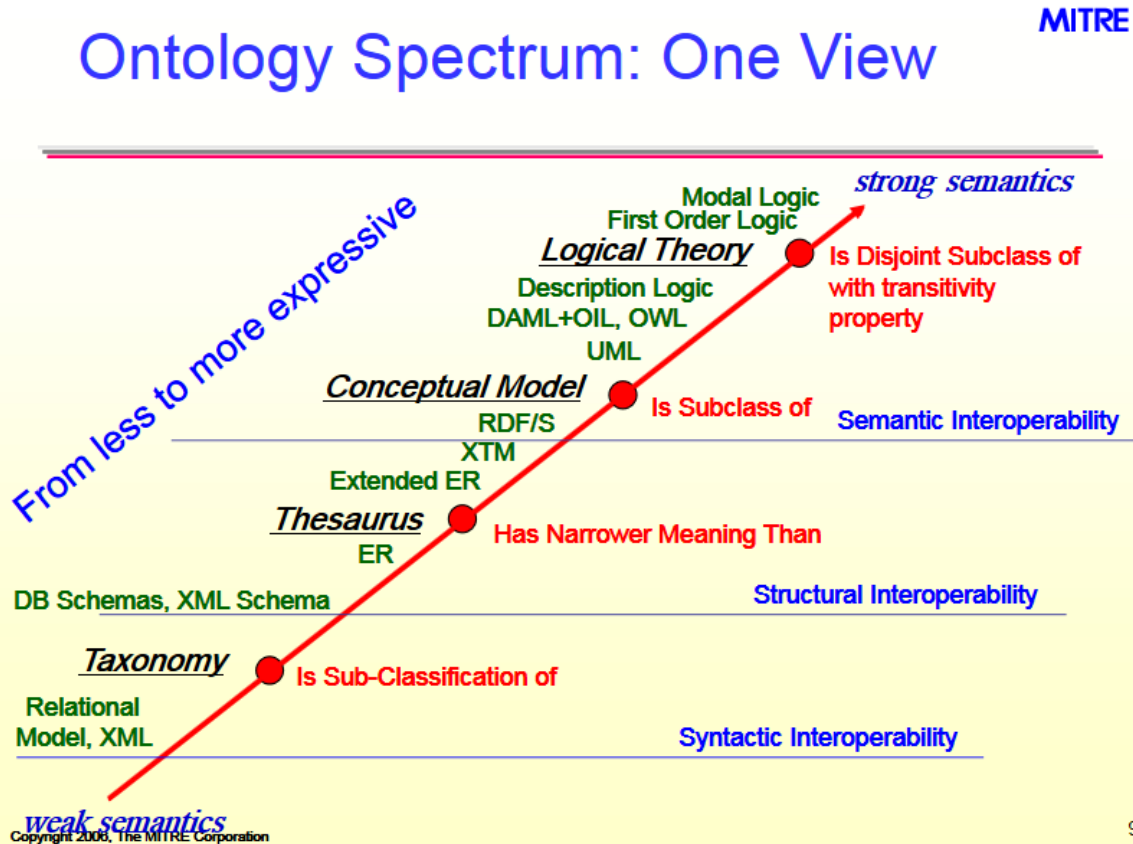
A representation language or framework which cannot be measured for simplicity is a recipe for unnecessary complexity. Conscientious knowledge engineers are compelled to express a problem domain’s conceptual model as richly as possible. With a highly-expressive language at a knowledge engineer’s disposal it is possible to think through different representational options at a level of detail that is impossible with a weaker-expressive language. Stronger languages push one more than one using a weaker language. Testing pushes one more than not using testing toward greater accuracy and comprehensiveness. As is said, “Ignorance is bliss.” Limitations of expressivity of the representation language used should be exposed so that the limitations become conscious.

### 1.2.25. Logic Representation Approaches

There are many different approaches that can be used to represent logic in machine readable form. Each approach has pros and cons. I have mentioned taxonomies and ontologies. There are entity relationship models, conceptual models, you can

use the Unified Modeling Language (UML) or a spinoff of UML called SysML<sup>104</sup>. The diagram above points out approaches such as RuleLog, Common Logic, Semantics of Business Vocabulary and Business Rules (SBVR), Z Notation. The one thing that each of these approaches have in common is that the typical business professional cannot understand any of these approaches.

Dr. Leo Obrst of Mitre published The Ontology Spectrum and Semantic Models<sup>105</sup> which provides the following slide:



The slide in general and the presentation in particular points out tools that are available for representing logic, or semantics, in machine readable form which can then be used by computer processes. Not all approaches are the same. Note the term "Logical Theory" and how high that approach is in terms of strong semantics. I will come back to that in a moment, but first three important ideas.

### 1.2.26. Understanding artificial intelligence and intelligent software agents

Artificial intelligence is the automation of activities that we associate with human thinking and activities such as decision making, problem solving, learning and so on.

An intelligent software agent<sup>106</sup> is software that assists people and acts on their behalf. Intelligent agents work by allowing people to:

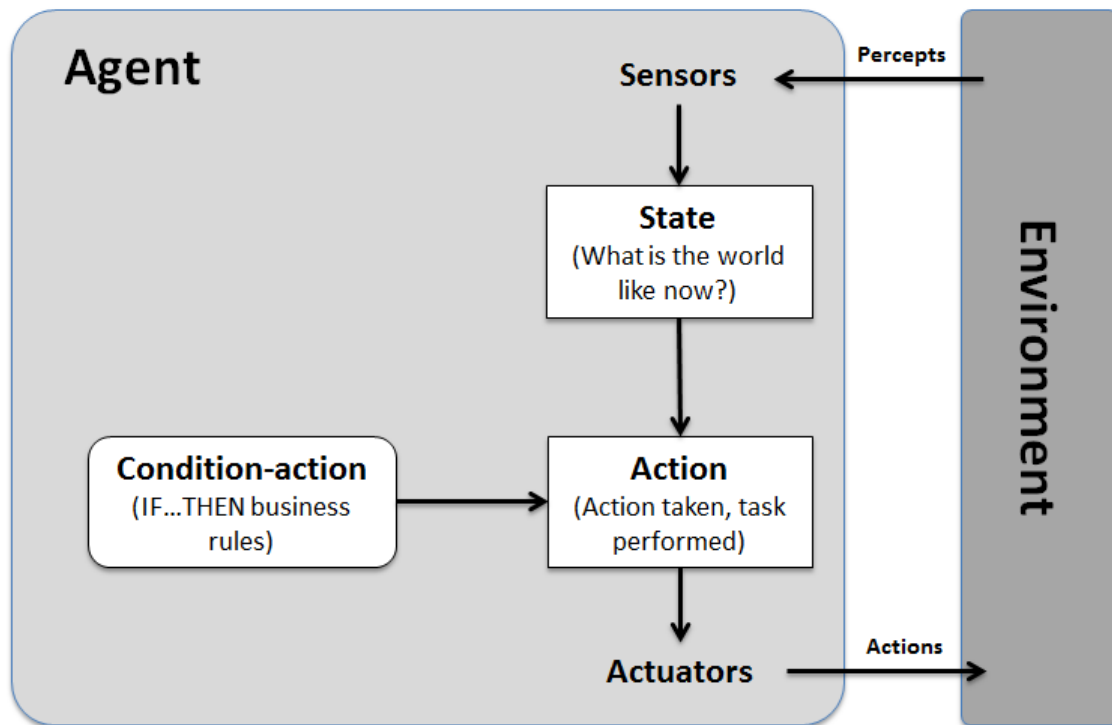
<sup>104</sup> SysML, <https://sysml.org/>

<sup>105</sup> Dr. Leo Obrst, *The Ontology Spectrum and Semantic Models*, slide 9, <https://slideplayer.com/slide/697642/>

- delegate work that they could have done to the agent software.
- perform repetitive tasks,
- remember things you forgot,
- intelligently find, filter and summarize complex information,
- customize information to your preferences,
- learn from you and even make recommendations to you.

An **agent** is an entity capable of **sensing** the **state** of its **environment** and **acting** upon it based on a set of specified **rules**. An agent performs specific tasks on behalf of another. In the case of software, an agent is a software program. There are many different types of intelligent software agents<sup>107</sup>.

## Simple Reflex Agent



The document *Comprehensive Introduction to Intelligent Software Agents for Professional Accountants* goes into significantly more detail on the topic of intelligent software agents.

<sup>106</sup> *Comprehensive Introduction to Intelligent Software Agents*, [http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.7\\_ComprehensiveIntroductionToIntelligentSoftwareAgents.pdf](http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.7_ComprehensiveIntroductionToIntelligentSoftwareAgents.pdf)

<sup>107</sup> Wikipedia, *Intelligent Agent*, Retrieved July 24, 2016; [https://en.wikipedia.org/wiki/Intelligent\\_agent](https://en.wikipedia.org/wiki/Intelligent_agent)

### 1.2.27. Understanding expert systems

Expert systems is a branch of artificial intelligence. Expert systems are computer programs that are built to mimic or simulate or emulate human behaviour and knowledge; expert systems are computer application that performs a task that would otherwise be performed by a human expert. Expert systems solve problems by reasoning about knowledge represented in machine-readable form as "IF...THEN" rules that the machine simply follows.

Computers really are not thinking, they are only mimicking or simulating or emulating human though by following a clearly laid out set of machine-readable instructions to perform some task.

Frank Puppe<sup>108</sup> explains in his book *Systematic Introduction to Expert Systems* 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.

### 1.2.28. Components of a knowledge based system

A software based expert system, also referred to as a knowledge based system, has four primary components:

- **Database of facts:** A database of facts is a set of observations about some current situation or instance. The database of facts is "flexible" in that they apply to the current situation. The database of facts is machine-readable. An XBRL instance is a database of facts.
- **Knowledge base:** A knowledge base is a set of universally applicable rules created based on experience and knowledge of the practices of the best domain experts generally articulated in the form of IF...THEN statements or a form that can be converted to IF...THEN form. A knowledge base is "fixed" in that its rules are universally relevant to all situations covered by the knowledge base. Not all rules are relevant to every situation. But where a rule is applicable it is universally applicable. All knowledge base information is machine-readable. An XBRL taxonomy is a knowledge base.
- **Rules processor/inference engine:** A rules processor/inference engine takes existing information in the knowledge base and the database of facts and uses that information to reach conclusions or take actions. The inference engine derives new facts from existing facts using the rules of logic. The rules processor/inference engine is the machine that processes the information.
- **Explanation mechanism:** The explanation mechanism explains and justifies how a conclusion or conclusions are reached. It walks you through which facts and which rules were used to reach a conclusion. The explanation

---

<sup>108</sup> Frank Puppe, *Systematic Introduction to Expert Systems, Knowledge Representations and Problem-Solving Methods*, page 11 (Note that you can read Parts I and II on Google Books here, <https://books.google.com/books?id=kKqCAAQBAJ>)



mechanism is the results of processing the information using the rules processor/inference engine and justifies why the conclusion was reached.

### 1.2.29. *Benefits of a knowledge based system*

Benefits from the use of expert systems or knowledge based systems include:

- **Automation:** elimination of routine, boring, repetitive, mundane, mechanical tasks that can be automated
- **Consistency:** computers are good at performing repetitive, mechanical tasks whereas humans are not; computers do not make mistakes and are good at repeating exactly the same thing each time
- **Diligence and tenacity:** computers excel at paying attention to detail; they never get bored or overwhelmed and they are always available and will keep doing their job until the task is complete with the same attention to detail
- **Reduced down-time:** computer based expert systems are tireless and do not get distracted
- **Availability:** computer based expert systems are always available simultaneously in multiple places at one time; you get quick response times and can replace absent or scarce experts
- **Training:** the best practices of the best practitioners can be available to those that are new to and learning about a domain of knowledge
- **Longevity and persistence:** computer based expert systems do not change jobs or retire so knowledge gathered by an organization can remain within that organization
- **Productivity:** computer based expert systems are cheaper than hiring experts and costs can be reduced at the same time that quality increases resulting in increased productivity
- **Multiple opinions:** Systems can integrate the view of multiple experts within a system and choose between the preferred view of multiple expert opinions in the same system
- **Objectivity:** computers apply the same inductive and deductive logic consistently; emotion and personal preferences can be eliminated where they should be eliminated

Financial report creation software of the future will be an expert system which operates similar to how CAD/CAM software for creating blueprints. Expert systems are described in more detail in the document *Comprehensive Introduction to Expert Systems*<sup>109</sup>.

### 1.2.30. *Strengths of computers*

Computers have four fundamental strengths<sup>110</sup>:

---

<sup>109</sup> *Comprehensive Introduction to Expert Systems*, [http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.6\\_ComprehensiveIntroductionToExpertSystems.pdf](http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.6_ComprehensiveIntroductionToExpertSystems.pdf)

<sup>110</sup> Andrew D. Spear, *Ontology for the Twenty First Century: An Introduction with Recommendations*; page 4, <http://ifomis.uni-saarland.de/bfo/documents/manual.pdf#Page=4>

- **Information storage:** Computers can store tremendous amounts of information reliably and efficiently.
- **Information retrieval:** Computers can retrieve tremendous amounts of information reliably and efficiently.
- **Information processing:** Computers can process stored information reliably and efficiently, mechanically repeating the same process over and over.
- **Ubiquitous information distribution:** Computers can make information instantly accessible to individuals and more importantly other machine-based processes<sup>111</sup> anywhere on the planet in real time via the internet, simultaneously to all individuals.

So how do you harness this power provided by computers?

### 1.2.31. Major obstacles to harnessing the power of computers

There are a number of major obstacles to harnessing the power of computers to perform work for business professionals within one department, in an organization or across an entire supply chain. These obstacles include<sup>112</sup>:

- **Business professional idiosyncrasies:** Different business professionals use different terminologies to refer to exactly the same thing.
- **Information technology idiosyncrasies:** Information technology professionals use different technology options, techniques, and formats to encode information and store exactly the same information. These options are impacted by fads, trends, arbitrary personal preferences, and other such factors.
- **Inconsistent domain understanding of and technology's limitations in expressing interconnections:** Information is not just a long list of facts, but rather these facts are logically, mechanically, mathematically interconnected and generally used within sets which can be dynamic and used one way by one business professional and some other way by another business professional or by the same business professional at some different point in time. These relations are many times more detailed and complex than the typical computer database can handle. Business professionals sometimes do not understand that certain relations even exist.
- **Computers are dumb beasts:** Computers don't understand themselves, the programs they run, or the information that they work with. Computers are "dumb beasts". What computers do can sometimes seem magical. But in reality, computers are only as smart as the metadata they are given to work with, the programs that humans create, and the data that exists in databases that the computers work with.

If two computers use the same information formats and other technology aspects but use different terminology or different information organization strategies, the two computers will find it difficult or even impossible to interoperate. If this is the case, the only way to cross the chasm between these two different computers is with

---

<sup>111</sup> Wired: *The Web is Dead: Long Live the Internet*, <http://xbri.squarespace.com/journal/2010/9/24/wired-the-web-is-dead-long-live-the-internet.html>

<sup>112</sup> Andrew D. Spear, *Ontology for the Twenty First Century: An Introduction with Recommendations*; page 4, <http://ifomis.uni-saarland.de/bfo/documents/manual.pdf#Page=4>

human intervention. Often this involves re-keying information. Saying this another way, in order for two computers to interoperate it is essential that every aspect including terminology, world view, information formats, instructions and so forth necessary to translate from one computer to the second computer must be explicitly provided.

### **1.2.32. Business rules prevent anarchy**

The Merriam-Webster dictionary defines anarchy<sup>113</sup> as “a situation of confusion and wild behavior in which the people in a country, group, organization, etc., are not controlled by rules or laws.” Business rules prevent information anarchy<sup>114</sup>.

Business rules<sup>115</sup> guide, control, suggest, or influence behavior. Business 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. Business rules help shape judgment, help make decisions, help evaluate, help shape behavior, and help reach conclusions.

Business rules arise from the best practices of knowledgeable business professionals. A business rule is a rule that 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 business 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 business rules is a choice of business professionals. The meaning of a business rule is separate from the level of enforcement someone might apply to the rule.

Business professionals interact with business rules every day and may not even realize it. Most business rules are in human readable form. But business rules can be represented in both human-readable form and machine-readable form. With the move to digital, more and more business rules are being represented in both human readable form and more importantly machine-readable form. Machine-readable business rules help automate processes which have been manual in the past.

The *Business Rules Manifesto*<sup>116</sup> does a good job of describing what a business rule is. Article 9; Of, By, and For Business People, Not IT People; points out the need for these business rules to be managed by business professionals:

- 9.1. Rules should arise from knowledgeable business people.
- 9.2. Business people should have tools available to help them formulate, validate, and manage rules.
- 9.3. Business people should have tools available to help them verify business rules against each other for consistency.

---

<sup>113</sup> Merriam-Webster, *Anarchy* definition, <http://www.merriam-webster.com/dictionary/anarchy>

<sup>114</sup> *Understanding that Business Rules Prevent Anarchy*, <http://xbrl.squarespace.com/journal/2016/7/15/understanding-that-business-rules-prevent-anarchy.html>

<sup>115</sup> *Comprehensive Introduction to Business Rules for Professional Accountants*, <http://xbrl.azurewebsites.net/2016/Library/ComprehensiveIntroductionToBusinessRulesForProfessionalAccountants.pdf>

<sup>116</sup> *Business Rules Manifesto*, <http://www.businessrulesgroup.org/brmanifesto.htm>

Business professionals are the ones who understand the problem domain. As such, business professionals are the ones who understand the business rules or relations between the things in their problem domain.

Another term for business rules is business logic.

### **1.2.33. Conceptual models, ontologies and theories describe systems**

A **domain** is some area of knowledge or activity. A domain is a system. Different domains tend to use different terminology to describe the same ideas. One domain may even have the same idea expressed using different terminology.

The term **ontology** has been used in philosophy for thousands of years going back to the father of formal logic, Aristotle<sup>117</sup> (400 B.C.). Ontology is defined as the study of the things and the relations between things that exist in reality. The goal of philosophical ontology is to provide deliberate, clear, coherent and rigorously worked out accounts of the basic structures found in reality.

In more current times, the term ontology has become prominent in the area of computer science and information science. In computer science the term ontology generally refers to the standardization of a terminology framework such that information repositories can be constructed. Ontologies used by philosophers like Aristotle were not machine-readable. Ontologies used by computers are machine-readable.

A **conceptual model**<sup>118</sup> is a representation of a system, made of the composition of concepts which are used to help people know, understand, or simulate a subject the model represents. The aim of a **conceptual model** is to express the meaning of terms and concepts used by domain experts to discuss a problem, and to find the correct relationships between different concepts.

A **theory** is a tool for understanding, explaining, and making predictions about a system. A theory describes absolutes. A theory describes the principles by which a system operates. A theory can be right or a theory can be wrong; but a theory has one intent: to discover the essence of some system.

A theory is consistent if its theorems will never contradict each other. Inconsistent theories cannot have any model, as the same statement cannot be true and false on the same system. But a consistent theory forms a conceptual model which one can use to understand or describe the system.

A *conceptual model, ontology, or theory* all provide frameworks which help to make conceptual distinctions, understand those distinctions, and organize ideas. Such frameworks overcome the four major obstacles of getting a computer system to perform work. They are three different ways of achieving the same thing.

### **1.2.34. Differences between ontologies, rules, and schemas**

Ontologies and business rules are overlapping rather than disjointed<sup>119</sup>. Ontologies and rules are both approaches to representing knowledge and each approach has

---

<sup>117</sup> Aristotle's epistemology, [http://en.wikipedia.org/wiki/Aristotle#Aristotle.27s\\_epistemology](http://en.wikipedia.org/wiki/Aristotle#Aristotle.27s_epistemology)

<sup>118</sup> Wikipedia, *Conceptual Model*, [https://en.wikipedia.org/wiki/Conceptual\\_model](https://en.wikipedia.org/wiki/Conceptual_model)

<sup>119</sup> Benjamin Groszof, *Survey of Knowledge Representations for Rules and Ontologies*, page 4, <http://coherentknowledge.com/wp-content/uploads/2013/05/Ontolog-Forum-talk-OF-surveyKR-20131024-BNG.pdf>

strengths and weaknesses<sup>120</sup>. The primary challenge relating to figuring out the best approach to use to represent knowledge, as pointed out by John Sowa in his paper *Fads and Fallacies about Logic*<sup>121</sup>, is not technical at all. Most issues relate to an even more formidable challenge related to fads, trends, arbitrary preferences, politics, fallacies, misinformation, and alternative competing standards.

Approaches to representing knowledge can be placed into three broad groups: *ontologies*, *rules*, and *schemas*. The purpose of each of these is to represent some problem domain or conceptual model of some sort. The following is a summary of the essence of each of these three approaches to representing knowledge:

- **Ontology:** An ontology is generally a subset of knowledge that is definitional in nature and focuses on defining terminology and categories/subcategories/properties of terms and the relations between the categories/subcategories/properties. An ontology is a collection of taxonomies. Ontologies are axiom-based.
- **Rules:** Rules are assertions or statements which have an IF...THEN format that have logical implication. The set of rules can be definitional in nature and therefore in essence ontological. A rule (or assertion, statement) can be TRUE or it can be FALSE. Rules are model-based.
- **Schema:** A schema is an outline, diagram, or model that tend to describe structure. To examples of schemas are a database schema which describes the structure of data in a database and XML Schema which is used to describe the structure of an XML document.

Too many people are in either the "ontology" camp or the "rules" camp or the "schema" camp to the exclusion of all other alternatives which tends to create silos and each camp acting dogmatic. This is not helpful to business professionals trying to solve problems practically.

Business professionals should focus on logic and the problem solving logic provided by a system rather than the implementation details.

### 1.2.35. Logical Theory

A **theory** is a formal statement of rules about some subject that describes and otherwise explains the nature of that subject. A theory describes some aspect of the world and tries to describe the principles by which that aspect of the world operates. A theory can be right or wrong, but a theory is characterized by its intent: the discovery of essence. A theory does not simplify. A theory describes absolutes. A successful theory can become a fact. A theory is a tool for contemplating something with an intent to gain insight or understanding.

**Logic** a set of principles that form a framework for correct reasoning. Logic is a process of deducing information correctly. Logic is about the correct methods that can be used to prove a statement is true or false. Logic tells us exactly what is meant. Logic allows systems to be proven.

In logic, a **statement** is a sentence that is either true or false. You can think of statements as pieces of information that are either correct or incorrect. And

---

<sup>120</sup> Silvie Spreeuwenberg, *The Semantic Web and the Business Rules Approach ~ Differences and Consequences*, <https://www.brcommunity.com/articles.php?id=b257>

<sup>121</sup> John F. Sowa, *Fads, Misinformation, Trends, Politics, Arbitrary Preferences, and Standards*, <http://www.ifsowa.com/pubs/fflogic.pdf>

therefore, statements are pieces of information that you apply logic to in order to derive other pieces of information which are also statements. Statements are basically rules.

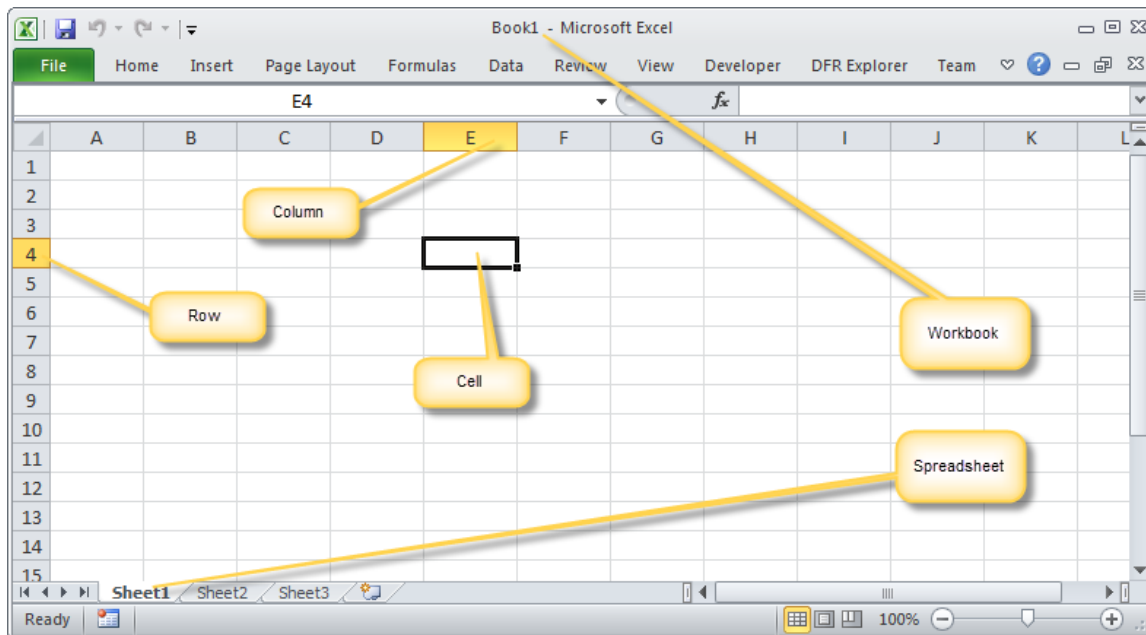
A **logical theory** is a set of logical statements that formally describes some subject or system. **Axioms**<sup>122</sup> are statements that describe self-evident logical principles that no one would argue with. **Theorems**<sup>123</sup> are logical deductions which can be proven by constructing a chain of reasoning by applying axioms and the rules of logic in the form of IF...THEN statements.

A **rule**, or business rule or assertion, is a true statement with respect to some model of the real world that could possibly exist given some logical theory. You cannot create rules that are true in worlds that can never exist. A rule can be a mathematical expression. A rule is a type of logical statement.

The **Financial Report Semantics and Dynamics Theory**<sup>124</sup> is a logical theory that explains how the mechanical aspects of a financial report work. The **Logical Theory Describing a Business Report**<sup>125</sup> explains how a general business report works. Both are based on the same ideas and tend to be significantly more understandable than other approaches for explaining the underlying logic of a system.

### 1.2.36. Using conceptual models

Business professionals work with conceptual models every day. For example, the workbooks, spreadsheets, rows, columns, and cells of an electronic spreadsheet are a conceptual model. The ease and simplicity of an electronic spreadsheet allows the average business professional to make use of this helpful tool.



<sup>122</sup> Wikipedia, *Axiom*, <https://en.wikipedia.org/wiki/Axiom>

<sup>123</sup> Wikipedia, *Theorem*, <https://en.wikipedia.org/wiki/Theorem>

<sup>124</sup> Charles Hoffman, CPA and Rene van Egmond, *Financial Report Semantics and Dynamics Theory*, <http://xbrlsite.azurewebsites.net/2016/Library/Theory-2017-06-26.pdf>

<sup>125</sup> Charles Hoffman, CPA and Rene van Egmond, *Logical Theory Describing a Business Report*, <http://xbrlsite.azurewebsites.net/2019/Library/LogicalTheoryDescribingBusinessReport.pdf>

### 1.2.37. Understanding the utility of the multidimensional model

Models help communication and provide a framework for understanding. The multidimensional model is a model for understanding information<sup>126</sup>. Every professional accountant works with multidimensional information every day and don't generally realize it.

Just like an electronic spreadsheet has a model (workbook, spreadsheet, row, column, cell); a digital financial report has a model. The model of a digital financial report follows the multidimensional model. Here are the high-level pieces of a digital financial report:

- **Fact:** A fact defines a single, observable, reportable piece of information contained within a financial report, or fact value, contextualized for unambiguous interpretation or analysis by one or more distinguishing characteristics. Facts can be numbers, text, or prose.
- **Characteristic:** A characteristic describes a fact (a characteristic is a property of a fact). A characteristic provides information necessary to describe a fact and distinguish one fact from another fact. A fact may have one or many distinguishing characteristics.
- **Fact table:** A fact table is a set of facts which go together for some specific reason. All the facts in a fact table share the same characteristics.
- **Relation:** A relation is how one thing in a business report is or can be related to some other thing in a business report. These relations are often called business rules. There are three primary types of relations (others can exist):
  - **Whole-part:** something composed exactly of their parts and nothing else; the sum of the parts is equal to the whole (roll up).
  - **Is-a:** descriptive and differentiates one type or class of thing from some different type or class of thing; but the things do not add up to a whole.
  - **Computational business rule:** Other types of computational business rules can exist such as "Beginning balance + changes = Ending Balance" (roll forward) or "Net income (loss) / Weighted average shares = Earnings per share".
- **Grain:** Grain is the level of depth of information or granularity. The lowest level of granularity is the actual transaction, event, circumstance, or other phenomenon represented in a financial report. The highest level might be a line item on a primary financial statement such as a balance sheet.

---

<sup>126</sup> *Introduction to the Multidimensional Model for Professional Accountants*, <http://xbrl.squarespace.com/journal/2016/3/18/introduction-to-the-multidimensional-model-for-professional.html>

### 1.2.38. *Understanding taxonomic keys*

A identification key or taxonomic key<sup>127</sup> is a printed or computer-aided device used for identifying some entity that is unknown. Keys are constructed so that the user is presented with a series of choices about the characteristics of the unknown thing; by making the correct choice at each step of the key, the user is ultimately led to the identity of the thing. Taxonomic keys are also helpful in classifying things into a standard taxonomy consistently.

There are two types of keys:

- **Single-access keys:** A single-access key (dichotomous key, sequential key, analytical key, or pathway key) is an identification key where the sequence and structure of identification steps is fixed by the author of the key.
- **Multi-access keys:** A multi-access key enables the user to freely choose the set and characteristics that are convenient to evaluate for the item to be identified.

Single-access keys and multi-access keys serve the same purpose. Each approach has its advantages and disadvantages.

One advantage of multi-access keys is that users can enter or select information about an unidentified thing in any order, allowing the computer to interactively rule out possible identifications of the entity and present the user with additional helpful information and guidance on what information to enter next. A disadvantage of multi-access keys is that you have to understand a certain amount about a domain to use them; the more you understand about a domain the more useful multi-access keys can be.

One advantage of single-access keys is that if you don't understand the domain or don't understand enough the single-access keys can serve as bread crumbs that provide a path to the answer you are looking for.

### 1.2.39. *Understanding prototype theory*

Fundamentally there are two perspectives to understanding what something is:

- Aristotle's perspective was "A thing is a member of a category if it satisfies the definition of the category."
- The second perspective, **prototype theory**<sup>128</sup>, is that we can know what something is even if it can't be clearly defined and even if its boundaries cannot be sharply drawn; concepts can be clear without having clear definitions if they're organized around undisputed examples, or prototypes.

---

<sup>127</sup> *Understanding Taxonomic Keys*, <http://xbrl.squarespace.com/journal/2016/5/13/understanding-taxonomic-keys.html>

<sup>128</sup> Wikipedia, *Prototype Theory*, Retrieved February 24, [https://en.wikipedia.org/wiki/Prototype\\_theory](https://en.wikipedia.org/wiki/Prototype_theory)



For example, one can understand that something is a “chair” by understanding as many properties as possible about the thing you are looking at, looking at the properties of a chair as defined by a prototype (the undisputed example), and then predicting whether the thing you are looking at is a “chair” by comparing the properties you are looking at with the properties of a chair.

By contrast, the definitional view “draws sharp lines” whereas the prototype view works because “things can be sort of, kind of in a category”. Prototype theory relies on our implicit understanding and does not assume that we can even make that understanding explicitly.

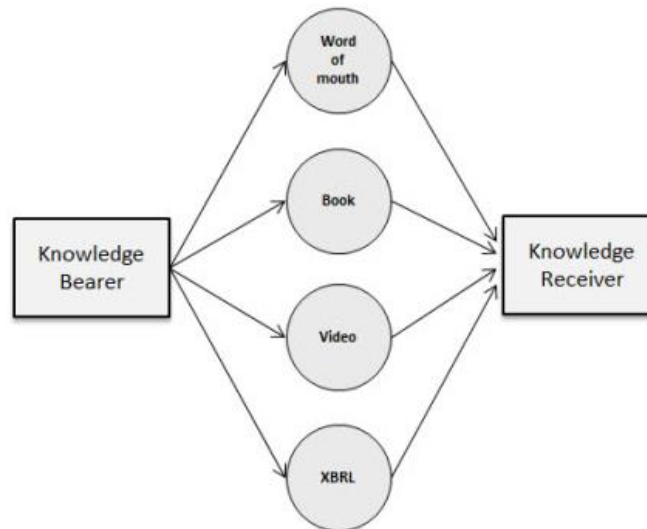
Computers need “handles” to be able to identify and work with things, see the section *Understanding the notion of identity*. If these handles are not provided, some other approach is necessary to work with something using a computer.

In addition to prototype theory, there is exemplar theory and multiple-prototype theory.

#### 1.2.40. Role of standards

The role of standards tends to be misunderstood and often under appreciated. These standards make things easier. There are many different standards such as intermodal shipping containers<sup>129</sup>, universal product codes, the metric system, JPEG photo format, MPEG audio format, etc.

Simply put, standards make things easier for users. XBRL is a global standard knowledge media<sup>130</sup>.



<sup>129</sup> Wikipedia, *Intermodal shipping container*, [https://en.wikipedia.org/wiki/Intermodal\\_container#Specifications](https://en.wikipedia.org/wiki/Intermodal_container#Specifications)

<sup>130</sup> *Understanding that XBRL is a Knowledge Media*, <http://xbrl.squarespace.com/journal/2017/1/16/understanding-that-xbrl-is-a-knowledge-media.html>

In order to make use of a knowledge media effectively, the following three conditions must be satisfied:

1. **Easy for knowledge bearer to represent information:** The effort and difficulty required for the knowledge bearer to successfully formulate the knowledge in the medium must be as low as possible.
2. **Clear, consistent meaning:** The meaning conveyed by the knowledge bearer to the knowledge receiver must be clear and easily followed by human beings and be consistent between different software applications. The result cannot be a "black box" or a guessing game and users of the information should not be able to derive different knowledge simply by using a different software application.
3. **High-quality information representation:** The form in which the knowledge is represented to the receiver must be as good as possible. The quality must be high whether the knowledge receiver is a human-being or an automated machine-based process. Sigma level 6 is a good benchmark, 99.99966% accuracy.

#### **1.2.41. Conceptual models serve the domain of the model**

There is not one best way to create a conceptual model. However, there are some common conceptual model design patterns that should be conscious in the minds of the creators of a conceptual model when trying to capture the knowledge of some domain. A conceptual model should be driven by the purpose of the conceptual model, the system or systems in which the conceptual model will be used, and the complexity and precision required by the domain and the system.

#### **1.2.42. "The map is not the territory"**

"The map is not the territory."<sup>131</sup> A map is not the territory it represents, but, if correct, the map has a similar structure to the territory, which accounts for its usefulness. The development of electronic media blurs the line between map and territory by allowing for the simulation of ideas as encoded in electronic signals.

Balance is everything. Everything which is simplistic is false. Everything which is complex is unusable. Simplicity is "dumbing down" a problem to make the problem easier to solve. Simple is about beating down complexity in order to make something simple and elegant; to make sophisticated things simple to use rather than complex to use.

Don't make the mistake of confusing the map and the territory.

### **1.3. Representing Domain Knowledge**

Given the idiosyncratic tendencies of business professionals; interpretations which reflect the arbitrary peculiarities of individuals can sometimes slip in or mistakes can be made when expressing such terminology. Further, parts of our understanding of a business domain can be incorrect and even evolve, improve, or simply change over time.

---

<sup>131</sup> Wikipedia, *Map-territory relation*, [https://en.wikipedia.org/wiki/Map%E2%80%93territory\\_relation](https://en.wikipedia.org/wiki/Map%E2%80%93territory_relation)

If different groups of business professional use different terminology for the same concepts and ideas to express the exact same truths about a business domain; those business professionals should be able to inquire as to why these arbitrary terms are used, identify the specific reasoning for this, and specifically identify concepts and ideas which are exactly the same as other concepts and ideas but use different terminology or labels to describe what is in fact exactly the same thing. But to also understand the subtleties and nuances of concepts and ideas which are truly different from other concepts and ideas.

If idiosyncrasies result only in different terms and labels which are used to express the exact same concepts and ideas; then mappings can be created to point out these different terms used to express those same concepts and ideas. Such mappings make dialogue more intelligible and could get groups to accept a single standardized term or set of terminology for the purpose of interacting with common repositories of business information.

If the difference in terminology and expression are rooted in true and real theoretical differences between business professionals, and the different terms express and point out real and important subtleties and nuances between what seemed to be the same terms; then these differences can be made conscious, explicit, clear, and therefore they can be discussed, in a rigorous and deliberate fashion because the differences are consciously recognized.

Business professionals must do this work. This work cannot be delegated to information technology professionals because they do not understand the subtleties and nuances of a business domain.

To perform this work, business professionals need to understand a few basic principles of representing knowledge in the form of a conceptual model, taxonomy, ontology, or theory.

### **1.3.1. Factual and heuristic knowledge**

An expert system's knowledge base contains two types of knowledge: factual and heuristic.

- *Factual knowledge* of a problem domain is knowledge that is widely shared and commonly agreed upon by those in the problem domain. You can generally find that knowledge in textbooks and journals.
- *Heuristic knowledge* of a problem domain is knowledge that tends to be more best practice and plausible reasoning for the domain. Heuristic knowledge is less rigorous and formal, more experiential in nature.

### **1.3.2. Understanding the notion of identity**

Computers need a way to grab onto information, a "handle" which software can use to identify something, grab it, and work with it. Things may or may not be unique. What does "us-gaap:StatementTable" mean if it is used to represent different things? To differentiate one use of us-gaap:StatementTable from another you need more information.

- **Isomorphic:** Has one meaning.
- **Polymorphic:** Has more than one meaning.

If no unique identity exists, a composite of multiple pieces of information might provide a unique identifier. For example, you need both the network and table to uniquely identify the fragments of an XBRL-based financial report submitted to the SEC by public companies. If unique identifiers are not available, then prototype theory can be used to identify the unique pieces of a financial report. (See the section *Understanding prototype theory* for more information.)

### **1.3.3. Differentiating a notion/idea/phenomenon, a name, and a preferred label**

It is important to understand and properly differentiate between the following three things:

- **Notion, idea, phenomenon:** something that exists in reality that needs to be represented
- **Name:** helps computers identify some notion/idea/phenomenon that is a representation of reality within some machine-readable conceptual model
- **Preferred label:** alternative ways used to refer to name

Confusing these three things can cause problems when trying to create a conceptual model. Two things that are genuinely different should have two different names. However, if one thing is given two names when the one thing really is two different preferred labels problems can occur.

For example, the FASB defines the notion of “Equity” in the US GAAP conceptual framework. The FASB defines “Equity”. The US GAAP XBRL Taxonomy defines the concept “us-gAAP:StockholdersEquity”. The FASB states specifically that “Net assets” is another preferred label for describing the notion of “Equity”. “Stockholders’ equity”, “Partner capital”, and “Proprietors’ equity” are all preferred labels for the notion of “Equity”.

### **1.3.4. Power of agreement**

It is only through deliberate, methodical, rigorous and conscious collaboration, cooperation and coordination by the participants of the financial reporting supply chain that XBRL-based digital financial reporting will work safely, reliably, predictably, repeatedly, effectively, and efficiently. This objective will not be achieved by accident.

Consider the definitions of arbitrary and standard:

- **Arbitrary:** based on random choice or personal whim, rather than any reason or system; depending on individual discretion (as of a judge) and not fixed by law
- **Standard:** used or accepted as normal; something established by authority, custom, convention, law, regulation, or general consent as a model or example

US GAAP contains many, many standard terms. For example, Equity, Assets, Liabilities, etc. The US GAAP XBRL Taxonomy names these terms, providing a standard.

A common obstacle to creating a working dictionary of concepts and relations between those concepts is disagreement as to those definition and relations. Agreement by all stakeholders through deliberate, methodical, rigorous and

conscious collaboration, cooperation, and coordination can help overcome this obstacle.

### ***1.3.5. Differentiating the important from the unimportant***

The following terms help one understand the difference between an important nuance and an unimportant negligible difference.

- **Nuance:** a subtle but important difference in or shade of meaning, expression, or sound; a subtle but important distinction or variation
- **Subtle:** so delicate or precise as to be difficult to analyze or describe but important; hard to notice or see but important; not obvious but important
- **Negligible:** so small or unimportant as to be not worth considering; insignificant; so small or unimportant or of so little consequence as to warrant little or no attention

Business professionals can best differentiate important nuances from unimportant negligible differences. They do not do it perfectly and the only real way to make sure things are right is testing and experimentation.

Conceptual models, ontologies, and theories are about getting the salient aspects of a problem domain right. One needs to take a pragmatic view of the world because it is impossible to describe every single aspect of the real world. Such frameworks only need to represent the important things and serve as a “wireframe” or a “substrate” of reality. Getting bogged down in unimportant, insignificant, or inconsequential details at best serves no purpose, at worst can cause unnecessary complexity.

### ***1.3.6. Difference between simplistic and simple***

Anyone can create something that is sophisticated and complex. It is much harder to create something that is sophisticated and simple. Simple is not the same thing as simplistic. “Simple” is not about doing simple things. Simple is the ultimate sophistication. Simple is elegant.

- **Simplicity:** Simplicity is “dumbing down” a problem to make the problem easier to solve. That is not what simple is about.
- **Simple:** Simple is about beating down complexity in order to make something simple and elegant; to make sophisticated things simple to use rather than complex to use.

Creating something that is simple takes conscious effort and is hard work.

### ***1.3.7. Difference between a requirement and a policy***

Sometimes things are required, other times things are a choice. Yet in other times setting some policy eliminates certain options which could have been previously considered.

- **Policy:** a course or principle of action adopted or proposed by a government, party, business, or individual; definite course or method of action selected from among alternatives and in light of given conditions to guide and determine present and future decisions

- **Requirement:** a thing that is needed or wanted; something that is essential or that must be done
- **Choice:** an act of selecting or making a decision when faced with two or more possibilities; the act of choosing; the act of picking or deciding between two or more possibilities
- **Option:** a thing that is or may be chosen; the opportunity or ability to choose something or to choose between two or more things

Any time a business professional is presented with an alternative; complexity increases because the business professional must choose. As the number of choices increases, complexity increases. Choices must be managed. Flexibility when it is not necessary is not a feature, it is a bug.

### ***1.3.8. Differentiating between objective and subjective***

There is a difference between something that is objective and something that is subjective.

- **Objective:** not influenced by personal feelings or opinions in considering and representing facts; based on facts rather than feelings or opinions; not influenced by feelings; facts are objective.
- **Subjective:** based on or influenced by personal feelings, tastes, or opinions; based on feelings or opinions rather than facts; relating to the way a person experiences things in his or her own mind; opinions are subjective.
- **Judgment:** the ability to make considered decisions or come to sensible conclusions; an opinion or decision that is based on careful thought; judgment is subjective.

Remember, computers are machines. Computers have no intelligence until they are instructed by humans. Computers only appear smart when humans create standards and agree to do things in a similar manner in order to achieve some higher purpose. It is easy to agree on things that tend to be objective. It is harder to agree where there is subjectivity. It is extremely difficult to impossible to get a machine to exercise judgment. A machine such as a computer can only mimic what humans tell the machine to do via machine-readable information.

### ***1.3.9. Difference between explicit and implicit***

In the process of agreeing, it is important to understand the difference between what is important and what is unimportant in that process of agreeing. It is likewise important to understand the difference between telling a machine something and requiring the machine to figure something out:

- **Explicit:** stated clearly and in detail, leaving no room for confusion or doubt; very clear and complete; leaving no doubt about the meaning.
- **Implicit:** implied though not plainly expressed; understood though not clearly or directly stated.
- **Ambiguous:** open to more than one interpretation; having a double meaning; able to be understood in more than one way; having more than one possible meaning; not expressed or understood clearly.

- **Derive or Impute:** assign (a value) to something by inference from the value of the products or processes to which it contributes; to deduce a conclusion about some fact using some other fact or facts and logical reasoning.

Machines do well with information which is explicitly provided. When information is not explicitly provided, software developers either make a choice or have to figure out ways to allow a business professional making use of the software to make a choice. Every time a software developer or business professional has to make an interpretation because something is ambiguous, there is the possibility that some unexpected or incorrect interpretation can be made. Not being explicit causes confusion and turns using ambiguous information into a guessing game.

### **1.3.10. Difference between assuming and deriving information**

Assuming new information and logically deriving new information from other information is not the same thing. When you make an assumption<sup>132</sup> you “take it upon oneself” or “lay claim to” or “take for granted” some fact or statement as being true. When you derive new information using logical processes, there is a logically supportable or logically defensible line of reasoning that (a) follow the rules of logic and (b) result in some new but supportable true fact or statement.

### **1.3.11. Conceptual models serve the domain of the model**

There is not one best way to create a conceptual model. However, there are some common conceptual model design patterns that should be conscious in the minds of the creators of a conceptual model when trying to capture the knowledge of some domain. A conceptual model should be driven by the purpose of the conceptual model, the system or systems in which the conceptual model will be used, and the complexity and precision required by the domain and the system.

Attempting to describe reality can lead to complex philosophical, theoretical, and even religious debates. But if one sees the goal as not to debate, but rather to be pragmatic and achieve something useful, the discussion becomes less complicated. One can distill the perspectives one might take to viewing reality into two possible extremes.

One approach to viewing reality is to take the perspective that reality (the world) exists objectively in-and-of itself; that *reality is independent of any one person*. Therefore, reality is knowable; the world exists and its properties are there to be discovered. This view implies that reality is objective and knowable and therefore constraints can exist as to what can be said about reality. In other words, conceptual models which provide representations of the world could get things wrong. Therefore, a conceptual model is right insofar as it accurately reflects the way the world is.

A second approach is to believe that *there is no one reality, that every individual perceives the world and that individual perception is reality*. This view implies that reality is completely subjective. This view does not imply that reality is not knowable because there are so many realities that it is impossible to agree on one reality. Rather, it implies that there are “reality camps” or groups of individuals with common beliefs about reality. Therefore, a conceptual model can represent one

---

<sup>132</sup> Merriam-Webster, Assumption, <https://www.merriam-webster.com/dictionary/assumption>

“reality camp”. That implies that a conceptual model can be created for each reality camp. Therefore, the second approach becomes equivalent to the first approach.

And so, a conceptual model can be created to represent each reality.

### **1.3.12. Pitfalls of knowledge engineering**

There are many different ways to stumble when trying to represent the knowledge of a problem domain in machine readable form. The following is a summary of many common pitfalls which should be recognized and then avoided.

#### ***One rigid reality***

Many of the things in a business problem domain are the invention of humans: the foot or meter, currency such as the US Dollar or the Euro, laws, regulations, accounting rules, concept of a legal entity. As such, to a large extent these things that are the creation of humans are malleable. At times there cannot be one single “correct” conceptual model for things in a problem domain because of inconsistencies in these human inventions. And so it can be the case that there is no single objectively correct answer, but possibly some set of pragmatically-based set of correct answers of some set of groups of users with clearly defined goals but with different sets of interests or self-interest of the specific set or group.

Fundamentally, excessive commitment to reality can lead to an inappropriate level of flexibility or inflexibility.

#### ***Overly complicated representation***

On the one hand, one must be careful of the illusion of clarity and apparent rigor where, in fact, there is little or no rigor or clarity. These illusions mask problems definitions of things which can be exceedingly difficult and even problematic to correctly characterize or how things interact with one another. Some problem domain things can be untenable regardless if one attempts to articulate the things in machine-readable form. Not recognizing such issues provides a false sense of meaningful information exchange.

Overly complicated representations are spots where the illusion of clarity can hide. Making things obscure by adding unnecessary and perhaps inaccurate details. This also adds to complexity which is simply not necessary.

#### ***Blind trust of domain experts***

Knowledge engineering calls for careful attention being paid to domain experts characterization of a domain by skilled knowledge engineers. But giving blind trust to domain experts is not appropriate. Knowledge engineers must have a critical side, analyzing and challenging representations for consistency and adequacy. Domain experts are not always right. Blind trust can lead to inappropriate tolerances and otherwise poorly constructed knowledge representations and ultimately an unworkable machine-readable representation.

One of the best ways to overcome this pitfall is to use deliberate and rigorous testing in order to check understanding.

#### ***Misuse of highly-expressive languages***

Using a highly-expressive language is no guarantee against sloppiness or process deficiencies. Highly-expressive languages offer the power and ability to articulate rich and precise rules for important classes and relations between classes. A weakly-



expressive language encourages sloppiness and commonly leads to inaccuracies due to the deficiencies in ability of the weakly-expressive languages to articulate important classes and relations between classes. Where only weak-expressivity is available rich expressiveness is not even available to the knowledge engineer; the result can be a superficial representation which is not useable by the problem domain.

### ***Recognize that pitfalls are avoidable***

Pitfalls are avoidable. Limitations are many times unavoidable and must be worked around. While the real world is malleable and there are always options for representing classes and relations between classes in various ways; this does not mean that everything can be created in any way one pleases. Using one approach in one specific area can mean that options are constrained for some other area of the representation. Dysfunctional, irrational, nonsensical, illogical, inconsistencies, and other issues which cause problems must be discovered and dealt with.

There is a difference between conscious inconsistencies and unconscious inconsistencies. Conscious inconsistencies are generally choices which are made because things are truly different, perhaps only subtle differences or nuances. Unconscious inconsistencies are generally due to sloppiness and lack of attention to detail and cannot be explained which pointed out and questioned.

### ***1.3.13. Rigorous testing maximizes communication and quality***

The best way of assuring that a machine-readable representation is not dysfunctional, irrational, nonsensical, illogical, inconsistent or has some other issue is comprehensive, thorough, deliberate, rigorous testing. Another is examining empirical evidence. Testing is a robust and pragmatic approach to checking understanding and determining if communication has taken place between domain experts, knowledge engineers, and software engineers who ultimately must implement software.

### ***1.3.14. Becoming conscious of the goal***

As Stephen R. Covey pointed out in his seminal work *Seven Habits of Highly Effective People*<sup>133</sup>, "Begin with the end in mind." You become conscious of what you need to do when you are conscious of the goal that you desire to achieve.

Prudence dictates that using financial information from a digital financial report not be a guessing game. It is only through conscious effort that the specific control mechanisms can be put in place to realize this intent.

The goal is a system that works safely, reliably, predictably, repeatedly, effectively, and efficiently.

Information technology professionals creating software must be able to create software which yields the same result when it would seem obvious to a business professional using software that the result, such as a query of basic information from a financial report, should be exactly the same even if different software applications are used. Different software applications providing different results when the results should be the same is not a desirable outcome.

---

<sup>133</sup> *Seven Habits of Highly Effective People, Habit 2*, <http://www.amazon.com/The-Habits-Highly-Effective-People/dp/1455892823>

Conscious and skillful execution using this approach can create digital financial reporting which is simple and elegant; and yet a sophisticated and powerful tool. Information in a digital financial reports must be deliberately created to be clear, consistent, logically coherent, and otherwise unambiguous to make sure the guessing game never takes place.

- Complete solutions are better than incomplete solutions
- Less expensive solutions are better than more expensive solutions
- Powerful solutions are better than simplistic solutions
- Easy to maintain solutions are better than hard to maintain solutions
- Easy to use solutions are better than hard to use solutions
- Good solution performance is better than poor solution performance
- More scalable solutions are better than less scalable solutions
- Standard solutions are better than proprietary solutions

#### **1.4. Other Useful Information**

The following is other helpful information. This information tends to be more advanced and therefore more challenging for business professionals to understand.

##### **1.4.1. Understanding life cycle of a conceptual model**

Just like many other things a conceptual model, ontology or theory has a life cycle. The paper *Towards ontology evaluation across the life cycle*<sup>134</sup> explains the problem of not understanding that life cycle and not being able to evaluate the quality of an ontology:

“Problem: Currently, there is no agreed on methodology for development of ontologies, and there is no consensus on how ontologies should be evaluated. Consequently, evaluation techniques and tools are not widely utilized in the development of ontologies. This can lead to ontologies of poor quality and is an obstacle to the successful deployment of ontologies as a technology.”

The paper points out that there are five aspects to the quality of ontologies which need to be evaluated: intelligibility, fidelity, craftsmanship, fitness, deployability.

The paper provides this diagram of the different stages of the ontology life cycle:

---

<sup>134</sup> Towards ontology evaluation across the life cycle, [http://www.researchgate.net/publication/260834360\\_Toward\\_Ontology\\_Evaluation\\_across\\_the\\_lifecycle](http://www.researchgate.net/publication/260834360_Toward_Ontology_Evaluation_across_the_lifecycle)

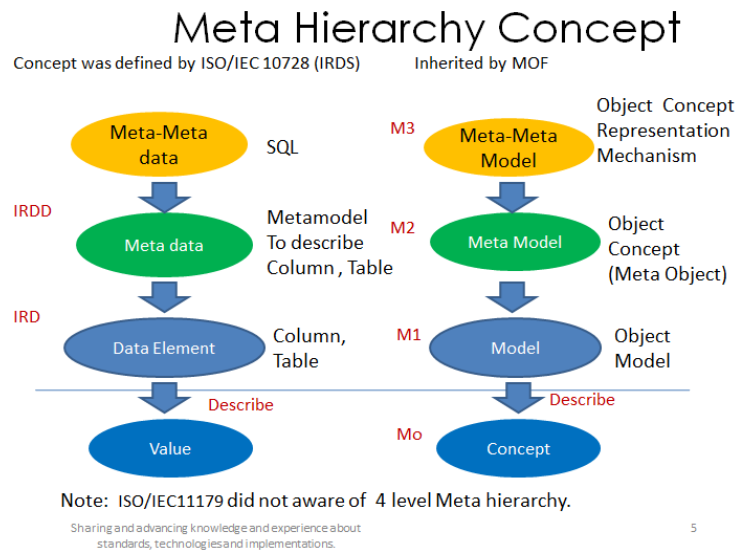


This is a list of the stages which are explained in the document: System design, Ontology design, Ontological analysis, Requirements definition, Operations/maintenance, Deployment, System development and integration, Ontology development and reuse.

These same ideas are appropriate and applicable to conceptual models, taxonomies, and theories.

**1.4.2. Conceptual model interoperability**

Different business domains and even different people in the same domain can create conceptual models differently. Yet, many times conceptual models need to interoperate. OMG<sup>135</sup> and ISO<sup>136</sup> have created a meta-meta model or hierarchy of concepts which are used to express conceptual models. This meta-meta model is intended to maximize interoperability between different conceptual models. This diagram provides an overview of that model.



<sup>135</sup> OMG Meta Object Facility, [https://en.wikipedia.org/wiki/Meta-Object\\_Facility](https://en.wikipedia.org/wiki/Meta-Object_Facility)

<sup>136</sup> ISO/IEC/IEEE 42010 Model and Metamodel, <http://www.iso-architecture.org/ieee-1471/meta/>

### 1.4.3. Reading list

The following books are extremely helpful in trying to knowledge engineering ideas. Anyone who wants to understand knowledge engineering in more detail should consider reading the following books to provide additional detail and build on the base of understanding derived from this conceptual overview:

**Data and Reality**<sup>137</sup>, by William Kent: (162 pages) While the first and last chapters of this book are the best, the entire book is very useful. The primary message of the Data and Reality book is in the last chapter, Chapter 9: Philosophy. The rest of the book is excellent for anyone creating a taxonomy/ontology and it is good to understand, but what you don't want to do is get discouraged by the detail and then miss the primary point of the book. The goal is not to have endless theoretical/philosophical debates about how things could be. The goal is to create something that works and is useful. A shared view of reality. That enable us to create a common enough shared reality to achieve some working purpose.

**Everything is Miscellaneous**<sup>138</sup>, by David Wenberger: (277 pages) This entire book is useful. This is very easy to read book that has two primary messages: (1) Every classification system has problems. The best thing to do is create a flexible enough classification system to let people classify things how they might want to classify them, usually in ways unanticipated by the creators of the classification system. (2) The big thing is that this book explains the power of metadata. First order of order, second order of order, and third order of order.

**Models. Behaving. Badly.**<sup>139</sup>, by Emanuel Derman: (231 pages) The first 100 pages of this book is the most useful. If you read the *Financial Report Semantics and Dynamics Theory*, you got most of what you need to understand from this book. But the book is still worth reading. It explains extremely well how it is generally one person who puts in a ton of work, figures something out, then expresses extremely complex stuff in terms of a very simple model and then thousands or millions of people can understand that otherwise complex phenomenon.

**Systematic Introduction to Expert Systems: Knowledge Representation and Problem Solving Methods**<sup>140</sup>, by Frank Puppe: (350 pages) The first three chapters of this book are an excellent introduction to expert systems, about 25 pages, and is easily understandable to a business professional. The second section of this book explains how expert systems work and the moving pieces of expert systems. The last two sections get technical, but are still understandable, and provide what amounts to an inventory of problem solving approaches and how to best implement those approaches in software.

**Semantic Web for the Working Ontologist**<sup>141</sup>, by Dean Allenmang and Jim Hendler: (354 pages) The first two chapters are the most useful. This is an extremely technical book, but the first chapter (only 11 pages) explains the big picture of "smart applications". It also explains the difference between the power of a query

---

<sup>137</sup> See, <http://xbrl.squarespace.com/journal/2014/7/28/data-and-reality-what-is-the-purpose-of-sec-xbrl-financial-f.html>

<sup>138</sup> See, <http://xbrl.squarespace.com/journal/2011/1/31/us-gaap-taxonomy-build-it-to-allow-reorganization.html>

<sup>139</sup> See, <http://xbrl.squarespace.com/journal/2014/7/20/updated-financial-report-semantics-and-dynamics-theory.html>

<sup>140</sup> See, <http://xbrl.squarespace.com/journal/2015/7/15/understanding-expert-systems-applicability-to-financial-repo.html>

<sup>141</sup> See, <http://www.amazon.com/Semantic-Web-Working-Ontologist-Effective/dp/0123735564>

language like SQL (relational database) and a graph pattern matching language (like XQuery). Querying can be an order of magnitude more powerful if the information is organized correctly.

***Ontology for the Twenty First Century: An Introduction with Recommendations***<sup>142</sup>, by Andrew D. Spear: (132 pages) The introduction first 45 pages are the best. This chapter is highly influenced by this resource. This can be challenging to make your way through but if you really want to understand all of the issues in creating useful ontologies; reading this is worth the effort.

**John Sowa**<sup>143</sup>. He has a wealth of knowledge.

## 2. Introduction to Problem Solving Logic

This section provides a comprehensive introduction to the notion of problem solving logic.

Some sort of approach or procedure is used to solve a problem. Said another way, solving a problem is not generally a “fly by the seat of your pants” or random effort. The best approaches to solving problems are deliberate and methodical.

The way a problem is solved is problem solving logic.

There is a difference between the ways a reasoning engine solves a problem and how a typical software program solves a problem. A reasoning engine is tuned to solve a specific set of problems. That means that it cannot solve other specific types of problems. A software program can be created to solve any specific problem. That means that to solve each type of problem, new code needs to be written. There are tradeoffs between an “engine” type of an approach and an “ad hoc” or roll-your-own type of an approach. Each approach works, but each has different sets of pros and cons.

Life is full of trade-offs. When evaluating available options the full cost and full benefits of each alternative need to be weighed in order to pick the alternative that best fits your needs.

Fads, misinformation, arbitrary preferences, ignorance, politics, trends, and other such things get in the way of making good decisions<sup>144</sup>. “Knowing one’s way about” brings great benefits. The philosopher Nicholas Rescher put it this way<sup>145</sup>:

“...Knowledge brings great benefits. The release of ignorance is foremost among them. We have evolved within nature into the ecological niche of an intelligent being. In consequence, the need for understanding, for ‘knowing one’s way about,’ is one of the most fundamental demands of the human condition.”

---

<sup>142</sup> See, <http://ifomis.uni-saarland.de/bfo/documents/manual.pdf>

<sup>143</sup> See, <http://www.jfsowa.com/logic/math.htm>

<sup>144</sup> John F. Sowa, *Fads, Misinformation, Trends, Politics, Arbitrary Preferences, and Standards*, <http://xbml.squarespace.com/journal/2016/9/23/fads-misinformation-trends-politics-arbitrary-preferences-an.html>

<sup>145</sup> Wikipedia, *Nicholas Rescher*, retrieved October 18, 2016, [https://en.wikipedia.org/wiki/Nicholas\\_Rescher](https://en.wikipedia.org/wiki/Nicholas_Rescher)

This document helps you know your way about and helps you work through the important area of problem solving logic. Why is this important? More and more information is becoming digital. For example, XBRL-based digital financial reports<sup>146</sup>.

To remain relevant in the digital age, professional accountants need to understand how computers solve problems.

## **2.1. Deconstructing the Notion of Problem Solving Logic**

In an interview with *Wired* magazine<sup>147</sup>, Barack Obama (yes, the ex-president of the United States discussing artificial intelligence) made the following statement about self-driving cars:

“There are gonna be a bunch of choices that you have to make, the classic problem being: If the car is driving, you can swerve to avoid hitting a pedestrian, but then you might hit a wall and kill yourself. It’s a moral decision, and who’s setting up those rules?”

This example which relates to self-driving cars points out two things that accounting professionals need to consider when thinking about XBRL-based digital financial reports: (1) who writes the rules, the logic, which software follows, (2) how do you write those rules and put them into machine readable form?

Computers work using the rules of mathematics. Mathematics works using the rules of logic. A problem solving logic is how a computer reasons.

To understand the notion of problem solving logic one first needs to understand the notion of logic and how logic can be applied to solving a problem. This section is dedicated to setting your perspective. The section provides specific definitions, deconstructing the pieces so that we can subsequently put the pieces back together.

### **2.1.1. Definition of a reasoning system**

Wikipedia defines a reasoning system<sup>148</sup> as “a software system that generates conclusions from available knowledge using logical techniques such as deduction and induction”.

The fact is, all computer systems are reasoning systems in that they all automate some type of logic or decision. For example, computing your annual income based on the number of hours worked and the pay rate per hour is reasoning. However, we want to talk about complete reasoning systems such as semantic reasoners or simply reasoner. Here is one definition of a semantic reasoner<sup>149</sup>:

“A semantic reasoner, reasoning engine, rules engine, or simply a reasoner, is a piece of software able to infer logical consequences from a set of asserted facts or axioms. The notion of a semantic reasoner generalizes that of an inference engine, by providing a richer set of mechanisms to work with. The inference rules are commonly specified by means of an ontology language,

---

<sup>146</sup> *Conceptual Overview of an XBRL-based, Structured Digital Financial Report*, <http://xbrlsite.azurewebsites.net/2016/Library/ConceptualOverviewOfDigitalFinancialReporting.pdf>

<sup>147</sup> *Wired, Barack Obama, Neural Nets, Self-driving Cars, and the Future of the World*, <https://www.wired.com/2016/10/president-obama-mit-joi-ito-interview/>

<sup>148</sup> Wikipedia, *Reasoning system*, retrieved October 18, 2016, [https://en.wikipedia.org/wiki/Reasoning\\_system](https://en.wikipedia.org/wiki/Reasoning_system)

<sup>149</sup> *Semantic Reasoner*, <http://hellosemanticweb.blogspot.com/2011/04/semantic-reasoners.html#axzz2URUuQy00>

and often a description language. Many reasoners use first-order predicate logic to perform reasoning; inference commonly proceeds by forward chaining and backward chaining.”

### 2.1.2. Definition of a logic

Logic is a formal system for enabling precise communication. A logic<sup>150</sup> is a set of principles underlying the arrangement of elements so as to perform some task. Merriam-Webster provides this simple definition of logic<sup>151</sup>:

- a proper or reasonable way of thinking about or understanding something
- a particular way of thinking about something
- the science that studies the formal processes used in thinking and reasoning

A logic is simply a set of rules and processes used to reason. Formal logic has been around since about 384 B.C. and was said to have been invented by Aristotle<sup>152</sup>. Logic is a discipline of philosophy. Logic is the study of correct reasoning.

The purpose of a logic is to communicate about some topic. Logic can be used to deduce complex principles from a commonly understood and agreed upon set of basic assumptions.

The same principles of logic work in all sciences and business domains, these logical principles are universal. One starts with a clearly stated and generally accepted set of hypotheses and perhaps some previously proven principles called theorems<sup>153</sup>. Each of these hypotheses and theorems state that if some situation occurs, then some other situation must also occur. Professional accountants understand this as IF...THEN statements.

### 2.1.3. Definition of a theory

A **theory**<sup>154</sup> is a prescriptive or normative statement which makes up a body of knowledge about what ought to be. A theory provides goals, norms, and standards. To theorize is to develop a body of knowledge.

A theory is a tool for understanding, explaining, and making predictions about a system. A theory describes absolutes. A theory describes the principles by which a system operates. A theory can be right or a theory can be wrong; but a theory has one intent: to discover the essence of some system.

A theory is consistent if its theorems will never contradict each other. Inconsistent theories cannot have any model, as the same statement cannot be true and false in the same system. But a consistent theory forms a conceptual model which one can use to understand or describe the system. A conceptual model or framework helps to make conceptual distinctions and organize ideas.

---

<sup>150</sup> Richard Hammack, Virginia Commonwealth University, *Book of Proof*, <http://www.people.vcu.edu/~rhammack/BookOfProof/BookOfProof.pdf>

<sup>151</sup> Merriam-Webster, *Logic definition*, retrieved October 18, 2016, <http://www.merriam-webster.com/dictionary/logic>

<sup>152</sup> Wikipedia, *Aristotle*, retrieved October 18, 2016, <https://en.wikipedia.org/wiki/Aristotle>

<sup>153</sup> Wikipedia, *Theorem*, retrieved June 20, 2017, <https://en.wikipedia.org/wiki/Theorem>

<sup>154</sup> Wikipedia, *Theory*, retrieved August 29, 2016, <https://en.wikipedia.org/wiki/Theory>

Theories are the real thing. A theory describes the object of its focus. A theory does not simplify. Theories are irreducible, the foundation on which new metaphors can be built. A successful theory can become a fact.

**Axioms** describe self-evident logical principles that no one would argue with. Axioms deal with primitives and fundamentals. An axiom is a premise so evident that it is accepted as true without controversy. **Theorems** are deductions which can be proven by constructing a chain of reasoning by applying axioms in the form of IF...THEN statements. A theorem is a statement that has been proven on the basis of previously established theorems or generally accepted axioms.

A **proof**<sup>155</sup>, or formal proof<sup>156</sup>, is a set of axioms and theorems that are used to determine if a *theory* is true.

The rules of logic are used to prove a theory<sup>157</sup>. Logic is sequences of reasoning for determining whether a set of axioms and theorems that form some theory are true or false.

#### 2.1.4. Putting logic into machine readable form

Description logics<sup>158</sup> are a family of formal knowledge representation logical languages. Description logics have varying levels of expressive power.

One important description logic is *SROIQ*<sup>159</sup>. The *SROIQ* description logic is the basis for the web ontology language, OWL 2 DL<sup>160</sup>. OWL 2 DL was designed to be implemented using software. Software can read OWL 2 DL and reason because of the knowledge represented in that machine-readable format. There are many other machine-readable ways of representing such information.

The expressiveness of OWL 2 DL and *SROIQ* description logic is limited; for example, mathematical relations cannot be expressed using that logical language. The point is, while such logics can be put into machine-readable form, not all logics are equivalent and not all problem solving logics are equivalent.

In the next section we will look at several different and powerful problem solving logics.

## 2.2. Fundamentals of logic

Professional accountants use logic informally every day. For example, something that you probably learned in school is BASE; beginning balance + additions – subtractions = ending balance. If you know any three facts, you can always find the fourth fact.

Here is an example of logic. Suppose you were trying to find the subtractions from some account.

---

<sup>155</sup> Richard Hammack, *Book of Proof*, <http://www.people.vcu.edu/~rhammack/BookOfProof/>

<sup>156</sup> Wikipedia, *Formal Proof*, retrieved October 18, 2016, [https://en.wikipedia.org/wiki/Formal\\_proof](https://en.wikipedia.org/wiki/Formal_proof)

<sup>157</sup> YouTube, *Crash Course in Formal Logic Part 1*, <https://www.youtube.com/watch?v=ywKZqjpMBUU>

<sup>158</sup> Wikipedia, *Description Logic*, retrieved October 18, 2016, [https://en.wikipedia.org/wiki/Description\\_logic](https://en.wikipedia.org/wiki/Description_logic)

<sup>159</sup> Ian Horrocks and Oliver Kutz and Ulrike Sattler, *The Even More Irresistible SROIQ*, <http://www.cs.ox.ac.uk/people/ian.horrocks/Publications/download/2006/HoKS06a.pdf>

<sup>160</sup> W3C, *OWL 2 Web Ontology Language Primer (Second Edition)*, <http://www.w3.org/TR/2012/REC-owl2-primer-20121211/>



1. You know that the beginning balance of the account is \$5,000
2. You know that the additions to the account was \$1,000
3. You know the ending balance of the account is \$2,000
4. You know that beginning balance + additions – subtractions = ending balance
5. You can derive the fact that the subtractions to the account are \$4,000 using the information provided in #1, #2, #3, and #4 and using the rules of logic

In this example you are using logic to reliably derive the subtractions from some account even though you do not know the actual value of the subtractions. And in doing so you are adding new information to your base of knowledge.

There is an important point to recognize here. Logic is the process of deducing information correctly; logic is not about deducing correct information. Consider the example above again. Suppose you were give incorrect information for the beginning balance of the account you are analyzing, that the beginning balance was \$9,000 rather than \$5,000 as stated above.

Our deduction that the subtractions are \$4,000 is now untrue. But the logic is perfectly correct; the information was pieced together correctly, even if some of that information was false.

Understanding the distinction between correct logic and correct information is important because it is important to follow the consequences of an incorrect assumption.

Ideally, we want both our logic to be correct and the facts we are applying the logic to, to be correct. But the point here is that correct logic and correct information are two different things.

If our logic is correct, then anything we deduce from such information will also be correct.

### **2.2.1. Logical systems**

A logical system<sup>161</sup> is an organization of terms and rules used for the analysis of information deduction. It consists of a language which is used to construct sentences and rules for deriving sentences.

Important properties that logical systems are:

- **Consistency** which means that statements or facts don't contradict one another.
- **Validity** which means that the system's rules never allow a false inference of a statement or fact from true premises.
- **Completeness** which means that if a rule is true, the rule can be proven and therefore the rule is justifiable.

---

<sup>161</sup> Wikipedia, *Logical Systems*, retrieved June 3, 2017, [https://en.wikipedia.org/wiki/Logic#Logical\\_systems](https://en.wikipedia.org/wiki/Logic#Logical_systems)

- **Soundness** which means that any information in the form of a statement or fact that is part of the system is true.

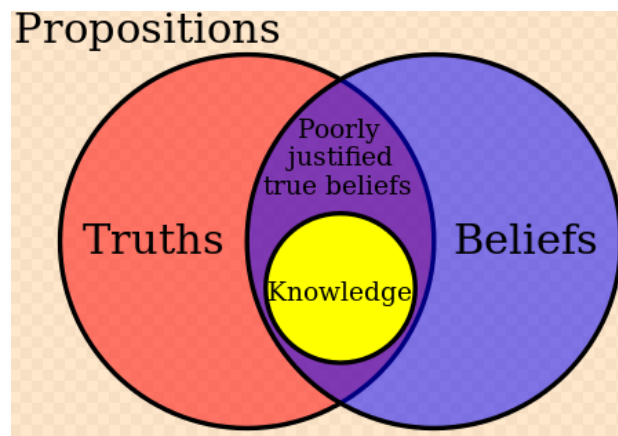
### 2.2.2. Knowledge is justified true belief

Knowledge is a familiarity, awareness, or understanding of someone or something, such as facts, information, descriptions, or skills, which is acquired through *experience* or *education* by perceiving, discovering, or learning<sup>162</sup>.

There are three minimum conditions for facts and information to be considered knowledge:

- The information must be **true**.
- You consciously **believe** that the information is true.
- **Justification** is present in the form of sufficient evidence to prove that the information is true. (i.e. a person has sufficient justification for believing what they believe)

The following is a diagram<sup>163</sup> which shows the intersection of what is true, what is believed to be true, and knowledge:



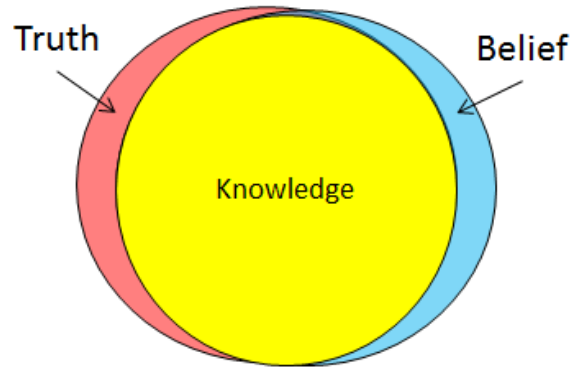
While philosophy likes to debate the meaning of knowledge, and while the majority agrees that knowledge is justified true belief; there are others that do not believe that knowledge is justified true belief. In fact, there is a famous example, the Gettier problem<sup>164</sup>, that shows a flaw in that definition of knowledge.

But what if you could create a logical system that has consistency, validity, completeness, and soundness and that the overlap between truth and belief is high providing the maximum amount of knowledge:

<sup>162</sup> Wikipedia, *Knowledge*, retrieved June 2, 2017, <https://en.wikipedia.org/wiki/Knowledge>

<sup>163</sup> Wikipedia, *Intersection of Knowledge*, retrieved June 2, 2017, [https://en.wikipedia.org/wiki/Belief#/media/File:Classical\\_definition\\_of\\_Kno.svg](https://en.wikipedia.org/wiki/Belief#/media/File:Classical_definition_of_Kno.svg)

<sup>164</sup> Wikipedia, *Gettier problem*, retrieved June 2, 2017, [https://en.wikipedia.org/wiki/Gettier\\_problem](https://en.wikipedia.org/wiki/Gettier_problem)



Such a system is creatable. Engineering is the application of a systematic, disciplined, quantifiable, methodical rigorous approach to the development, operation, and maintenance of something. Those who are skilled in logic can build such systems and such systems are very useful.

### 2.2.3. Sets

A set<sup>165</sup> is a well-defined collection of objects. Sets have members. The criteria for membership in a set must be well understood. Three set operations are important to understand:

- **Union:** All objects that are in set "A" plus all the objects in set "B" (i.e. objects that are either in set A or in set B or in both sets A and B).
- **Intersection:** All objects that are common to both sets "A" and "B".
- **Complement:** All objects that are members of set "A" but NOT members of set "B".

The notion and rules of sets can be leveraged by logical systems. For example, relational databases leverage set theory.

### 2.2.4. Logical statements

In logic, a statement is a sentence that is either true or false. You can think of statements as pieces of information that are either correct or incorrect. And therefore, statements are pieces of information that you apply logic to in order to derive other pieces of information which are also statements.

Here are some examples of statements:

- "Assets = Liabilities and equity", i.e. the accounting equation<sup>166</sup>.
- "Beginning balance + additions – subtractions = ending balance", i.e. as stated above and the definition of a roll forward.
- "Originally stated balance + adjustments = Restated balance"

<sup>165</sup> Wikipedia, *Set*, retrieved June 20, 2017, [https://en.wikipedia.org/wiki/Set\\_\(mathematics\)](https://en.wikipedia.org/wiki/Set_(mathematics))

<sup>166</sup> Wikipedia, *Accounting Equation*, retrieved June 2, 2017, [https://en.wikipedia.org/wiki/Accounting\\_equation](https://en.wikipedia.org/wiki/Accounting_equation)

- Some types of revenue are operating.
- Assets for the consolidated entity as of the balance sheet date of December 31, 2016 is \$45,000.

A financial report discloses facts. One fact in a report is distinguishable from another fact in the report by the characteristics of each of the two facts. One fact in a report can be related to another fact in a report and the relation between those two facts should be consistent with expectations which are established by rules.

Logic is about the correct methods that can be used to prove that a statement is true or false. To prove that a statement is true, we start with statements other statements that are proven to be true and use logic to deduce more and more complex statements until finally we obtain a statement that we are looking to determine if the statement is true or false. Of course some statements are more difficult to prove than others; in this resource we will concentrate on statements that are easier to prove to help you learn the basics of logic. But the point is this: In proving that statements are true, we use logic to help us understand statements and to combine pieces of information to produce new pieces of information.

As your skills grow you can create increasing complex statements.

A financial report is complex logical information<sup>167</sup>. XBRL-based financial reports are machine-readable structured information.

### **2.2.5. Conditional statements**

Another way to represent statements is in the IF...THEN format. These are called conditional statements. Here is an example of a conditional statement<sup>168</sup>.

- IF Assets = Liabilities and Equity; THEN the balance sheet balances.

Note that you may not be able to reverse a conditional statement. For example, the above conditional statement reversed would be:

- IF the balance sheet balances; THEN Assets = Liabilities and Equity.

The "IF" part of a conditional statement is called the *hypothesis* and the "THEN" part is called a *conclusion*. There are other terms that are used such as the *antecedent* and *consequent*.

### **2.2.6. Truth tables**

A truth table helps you figure out whether a statement is TRUE or FALSE, generally conditional statements.

---

<sup>167</sup> Charles Hoffman, *Processing Complex Logical Information or Structured Knowledge*, <http://xbrl.squarespace.com/journal/2017/4/22/processing-complex-logical-information-or-structured-knowled.html>

<sup>168</sup> Wikipedia, *Material Conditional*, retrieved June 2, 2017, [https://en.wikipedia.org/wiki/Material\\_conditional](https://en.wikipedia.org/wiki/Material_conditional)

Here is an example of a truth table:

<i>Assets Exists</i>	<i>Liabilities and Equity Exists</i>	<i>Assets exists AND Liabilities and Equity Exists</i>	<i>NOT (Assets Exists AND Liabilities and Equity Exists)</i>	<i>NOT (Assets Exists)</i>	<i>NOT (Liabilities and Equity Exists)</i>	<i>NOT (Assets exists OR Liabilities and Equity Exists)</i>
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

### 2.2.7. Logical equivalence

Logical equivalence<sup>169</sup> means that two statements mean *exactly* the same thing, they are logically equivalent, they have the same logical content.

### 2.2.8. Logical contradiction

A logical contradiction is a statement that is false.

### 2.2.9. Logical operators

Logic has the following fundamental low-level logical operators:

- Logical **AND**
- Logical **OR**
- Logical **NOT**
- Logically **equivalent to**
- Logically **NOT equivalent to**

Logical operators are used to combined two statements to form some new statement. These low-level logical operators are used to create higher-level logical operators such as:

- Add (+)
- Subtract (-)
- Multiply (\*)
- Divide (/)

And they can also be used to create relational operators:

- Greater than (>)
- Less than (<)
- Equal to (=)

The point here is that the basic low-level logical operators are the foundation of all problem solving logic<sup>170</sup>. There are different syntaxes of logical operators<sup>171</sup>, such a

<sup>169</sup> Wikipedia, *Logical Equivalence*, retrieved June 2, 2017, [https://en.wikipedia.org/wiki/Logical\\_equivalence](https://en.wikipedia.org/wiki/Logical_equivalence)

Z Notation. And logical operators can be used for many different things such as designing electrical circuits<sup>172</sup>. But all the low-level logical operators are the same.

### 2.2.10. Types of reasoning

The following is a summary of the approaches to reasoning which can be used:

- **Deductive or direct reasoning:** Deductive reasoning is the process of finding a direct chain of reasoning, IF...THEN statements, that explicitly bridge some gap between what you know and your hypothesis.
- **Inductive or indirect reasoning:** Inductive reasoning, also known as reasoning by contradiction or *reductio ad absurdum* (reduced to absurdity), is the process of proving that all known alternatives to your hypothesis are not true and therefore your hypothesis must be true because all other hypothesis are not true and you cannot prove that your hypothesis is false.

Another term for inductive or indirect reasoning is logical inference.

### 2.2.11. Logical inference

Logical inference is the process of deriving new information from one or more existing pieces of information, deducing a conclusion about that information using the rules of logic. For example,

- Suppose you know that Assets = \$2,000
- Suppose you know that Current assets = \$500
- Suppose you know that Assets = Current assets + Noncurrent assets

Using the information provided above, you can use the rules of logic to derive the value of Noncurrent assets to be \$1,500 because Assets (\$2,000) = Current assets (\$500) + Noncurrent assets (UNKNOWN); but using the rules of math you solve for the value of the UNKNOWN; Assets (\$2,000) – Current Assets (\$500) = Noncurrent assets (UNKNOWN); finally you get to Noncurrent assets = \$1,500.

You are not guessing. You are using logic to determine, accurately, what the value of Noncurrent Assets is based on other facts that you know.

### 2.2.12. Final thoughts about logic

It is important to understand why you want to take the time to understand logic. Here are the primary reasons:

- Applying logic correctly helps you understand what something means.

---

<sup>170</sup> Charles Hoffman, *Comprehensive Introduction to Problem Solving Logic*, [http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.5\\_ComprehensiveIntroductionToProblemSolvingLogic.pdf](http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.5_ComprehensiveIntroductionToProblemSolvingLogic.pdf)

<sup>171</sup> Jonathan Jacky, *Glossary of Z Notation*, <https://staff.washington.edu/jon/z/glossary.html#logic>

<sup>172</sup> Updated by Richard Bigwood, *Basic Gates and Functions*, <http://www.ee.surrey.ac.uk/Projects/CAL/digital-logic/gatesfunc/>

- The rules of inference provide a system which can be used to reliably derive new information from existing information.
- Logic helps you not only understand the meaning of statements; it also helps you reliably produce new meaningful statements.

Logic is the glue that holds sets of statements together and helps you understand the exact, precise meaning of statements. Logic is a common language that can be agreed upon which enables communication.

Logic helps you get to true and unambiguous meaning and helps you reconcile your true and unambiguous meaning to the true and unambiguous meaning of others. Computers are machines that work using the rules of logic.

Logic helps to keep everyone on the same page.

### ***2.2.13. Subjectivity and professional judgement***

Accounting and financial reporting allow for subjectivity and professional judgement. This is not inconsistent with logic. Professional judgement allows an accountant to pick between allowed alternatives. Professional judgement allows an accountant to determine if something is material. Professional judgement allows for other sorts of subjectivity. However, there is no professional judgement when it comes to things such as whether a roll up actually rolls up or a roll forward actually rolls forward. Roll ups, roll forward, and many other logical, mechanical, and mathematical relations are truths. These higher-level truths are not open to interpretation and not subject to professional judgement.

Not everything in accounting and financial reporting is subjective. Only some things are. Correctly separating what is subjective from what is objective is critical. There are times where this can be ambiguous because of the rules of US GAAP. US GAAP was not designed to be ambiguous, but it was designed to allow for professional judgement.

## ***2.3. Understanding the Notion of Problem Solving Logic***

Creating a problem solving logic is a balancing act. You want the logic to have the maximum in terms of expressiveness. But you want the logic to be safely implementable in software application so that logical catastrophes do not occur which cause systems to crash or provide results that are not reliable or predictable.

### ***2.3.1. Describing systems formally***

Deliberate, rigorous, conscious, skillful execution is preferable to haphazard, negligent, unconscious, inept execution if you want to be sure something works. Engineering a system to make sure it works as designed is a very good thing.

Knowledge engineering is the process of representing information in machine-readable form<sup>173</sup>.

A digital financial report<sup>174</sup> is a type of formal system. Many aspects of a digital financial report are mechanical and those mechanical aspects of how such a report works can be described using a conceptual model. The *Financial Report Semantics and Dynamics Theory*<sup>175</sup> describe the conceptual model of a digital financial report.

A system such as the digital financial report needs to be described precisely so that professional accountants understand the mechanics of how the system works so that the system can be used effectively and so the system works how the system was intended to work. There are many tools that can be used to describe a system.

**Z Notation**<sup>176</sup> is an ISO/IEC standard for describing systems precisely. Z Notation is used to describe safety-critical systems such as nuclear power plants, railway signaling systems, and medical devices. But while Z Notation is precise, Z Notation is not machine-readable.

**Common Logic**<sup>177</sup> (CL), also an ISO/IEC standard, is a framework for a family of logic languages, based on first-order logic, intended to facilitate the exchange and transmission of knowledge in computer-based systems. Common Logic is machine-readable. Further, the logic allowed to be expressed by Common Logic is consciously limited to avoid logical catastrophes<sup>178</sup> which cause systems to break.

Common Logic is about being practical, something business professionals generally tend to like. Common logic is a conscious compromise in order to achieve reliability, predictability, and safety. Common Logic is a "sweet spot" that achieves high expressivity but consciously gives up certain specific things that lead to catastrophic results that cause systems to potentially break making a system unsound; so that a system will be sound. Common Logic establishes well-thought-out boundaries, allowing creators of systems to "stay within the lines" and if you do, you get a maximum amount of expressiveness with the minimum risk of catastrophic system failure. Thus, you get a more reliable, dependable system.

**Semantics of Business Vocabulary and Business Rules**<sup>179</sup> (SBVR) is an OMG standard that was designed and built to be logically equivalent to ISO/IEC Common Logic.

---

<sup>173</sup> *Comprehensive Introduction to Knowledge Engineering Basics for Professional Accountants*, <http://xbrl.azurewebsites.net/2016/Library/ComprehensiveIntroductionToKnowledgeEngineeringForProfessionalAccountants.pdf>

<sup>174</sup> *Conceptual Overview of an XBRL-based, Structured Digital Financial Report*, <http://xbrl.azurewebsites.net/2016/Library/ConceptualOverviewOfDigitalFinancialReporting.pdf>

<sup>175</sup> *Financial Report Semantics and Dynamics Theory*, <http://xbrl.squarespace.com/fin-report-sem-dyn-theory/>

<sup>176</sup> *Understanding the Importance of Z Notation*, <http://xbrl.squarespace.com/journal/2015/9/4/understanding-the-importance-of-z-notation.html>

<sup>177</sup> *Understanding Common Logic*, <http://xbrl.squarespace.com/journal/2016/6/23/understanding-common-logic.html>

<sup>178</sup> *Brainstorming the Idea of Logical Catastrophes or Failure Points*, <http://xbrl.squarespace.com/journal/2015/7/25/brainstorming-idea-of-logical-catastrophes-or-failure-points.html>

<sup>179</sup> OMG, *Semantics of Business Vocabulary and Business Rules (SBVR)*, section 2.5 Conformance of an SBVR Processor, page 7, <http://www.omg.org/spec/SBVR/1.0/>



**Rulelog**<sup>180</sup> is a logic that is consciously engineered to be consistent with ISO/IEC Common Logic and OMG Semantics of Business Vocabulary and Business Rules. Rulelog is a dialect of W3C's RIF<sup>181</sup>. RuleML<sup>182</sup> is a syntax for implementing rules.

**SHACL**<sup>183</sup> (Shapes Constraint Language) is a logic that is consciously engineered to work in a "closed world" similar to a relational database. SHACL is a W3C recommendation. SHACL is a language for validating RDF graphs against a set of conditions. SHACL is used to perform closed-world constraint checks on RDF-based data.

Other standard and proprietary syntaxes exist for implementing rules. What is the point? Ask yourself why ISO/IEC and OMG would go through the trouble to create specifications such as Z Notation, Common Logic, and Semantics of Business Vocabulary and Business Rules? The answer to that question is to enable systems to be described precisely so that they can be implemented successfully using computer software.

Information technology professionals will likely never agree on which specific implementation syntax is best. As such, there will likely be multiple technology stacks. But, the different technology stacks should be logically interoperable<sup>184</sup>. Multiple independent, but logically interoperable, stacks of languages; and the XBRL stack should be an option.

Logics can be used to describe systems. Standard logics, such as Common Logic and Semantics of Business Vocabulary and Business Rules, RuleLog, and SHACL enable interoperability. As John F. Sowa put it in *Fads and Fallacies about Logic*<sup>185</sup>:

"In summary, logic can be used with commercial systems by people who have no formal training in logic. The fads and fallacies that block such use are the disdain by logicians for readable notations, the fear of logic by nonlogicians, and the lack of any coherent policy for integrating all development tools. The logic-based languages of the Semantic Web are useful, but they are not integrated with the SQL language of relational databases, the UML diagrams for software design and development, or the legacy systems that will not disappear for many decades to come. **A better integration is possible with tools based on logic at the core, diagrams and controlled natural languages at the human interfaces, and compiler technology for mapping logic to both new and legacy software.**"

The bottom line is that the best balance between expressive power and safe implementation has been achieved by the ISO/IEC global standard Common Logic. Common Logic<sup>186</sup> is a framework for a family of logic languages, based on first-order logic, intended to facilitate the exchange and transmission of knowledge in computer-based systems. That safely expressive sweet spot is also used by the OMG

---

<sup>180</sup> Rulelog, <http://ruleml.org/rif/rulelog/spec/Rulelog.html>

<sup>181</sup> W3C, RIF Overview (Second Addition), <http://www.w3.org/TR/rif-overview/>

<sup>182</sup> RuleML, [http://wiki.ruleml.org/index.php/RuleML\\_Home](http://wiki.ruleml.org/index.php/RuleML_Home)

<sup>183</sup> W3C, Shapes Constraints Language, <http://www.w3.org/TR/shacl/>

<sup>184</sup> Michael Kifer, Jos de Bruijn, Harold Boley, and Dieter Fensel, *A Realistic Architecture for the Semantic Web*, <http://www3.cs.stonybrook.edu/~kifer/TechReports/msa-ruleml05.pdf>

<sup>185</sup> John F. Sowa, *Fads and Fallacies about Logic*, page 6, <http://www.jfsowa.com/pubs/fflogic.pdf>

<sup>186</sup> *Understanding Common Logic*, <http://xbrl.squarespace.com/journal/2016/6/23/understanding-common-logic.html>

standard Semantics of Business Vocabulary and Business Rules<sup>187</sup> which was consciously designed to be logically equivalent to ISO/IEC Common Logic.

The most important thing to realize is that there is a good, safe target in terms of an expressive logic that is also safely implementable in software so catastrophic failures are avoided. Another very good thing is that business professionals don't need to understand the underlying technical details of these logic standards, nor will they ever have to deal with them. Higher level languages that follow the foundations set by Common Logic, Semantics of Business Vocabulary and Business Rules, Rulelog, and SHACL.

### **2.3.2. XBRL is a problem solving logic that should be equivalent to Common Logic, SBVR, RuleLog and SHACL**

The XBRL technical syntax is a global standard logic for representing knowledge. While much of the logic such as XBRL elements, relations between elements, mathematical relations between concepts and facts (XBRL calculation relations and XBRL Formula relations), dimensional relationships between concepts and facts, and other such relations (expressible using XBRL definition relations); not all such relation logic is standard.

XBRL Formula processors have specific deficiencies in their processing capabilities<sup>188</sup>. To overcome these deficiencies, the following capabilities must exist or need to be added to XBRL Formula Processors:

- Support **normal global standard functionality** that high-quality XBRL Formula processors support (i.e. Arelle, UBmatrix/RR Donnelley, Fujitsu, Reporting Standards, etc.)
- **Support inference** (i.e. deriving new facts from existing facts using logic, what inference engines do)
- Improved support validation and use of **structural relations** (i.e. XBRL Taxonomy functions; this was consciously left out of the XBRL Formula specification in order to focus on XBRL instance functionality)
- Support **forward chaining** and possibly also backward chaining in the future (i.e. chaining was also proposed but was left out of the XBRL Formula specification)
- Support a **maximum amount of Rulelog logic** which is safely implementable and is consistent with ISO/IEC Common Logic and OMG Semantics of Business Vocabulary and Business Rules
- **Additional XBRL definition arcoles** that are necessary to articulate the Rulelog logic, preferably these XBRL definition relation arcoles would end up in the XBRL International Link Role Registry and be supported consistently by all XBRL Formula processors (i.e. these general arcoles, and these financial disclosure related arcoles; this human readable information is helpful to understand the arcoles)

---

<sup>187</sup> OMG, *Semantics of Business Vocabulary and Business Rules (SBVR)*, section 2.5 Conformance of an SBVR Processor, page 7, <http://www.omg.org/spec/SBVR/1.0/>

<sup>188</sup> Specific Deficiencies in Capabilities of Existing XBRL Formula Processors, <http://xbrl.squarespace.com/journal/2016/9/26/specific-deficiencies-in-capabilities-of-existing-xbrl-formu.html>

While added functionality might not be global standard functionality, the functionality is necessary to prove the logic of US GAAP based financial reporting or IFRS based financial reporting. US GAAP and IFRS semantics are relatively clear. What is not clear to some business professionals is how to convey that meaning using the XBRL global standard. Proprietary techniques for applying XBRL can be used to fill any gap. However, the logical rules used by any proprietary techniques should follow the logic of Common Logic, SBVR, RuleLog, and SHACL.

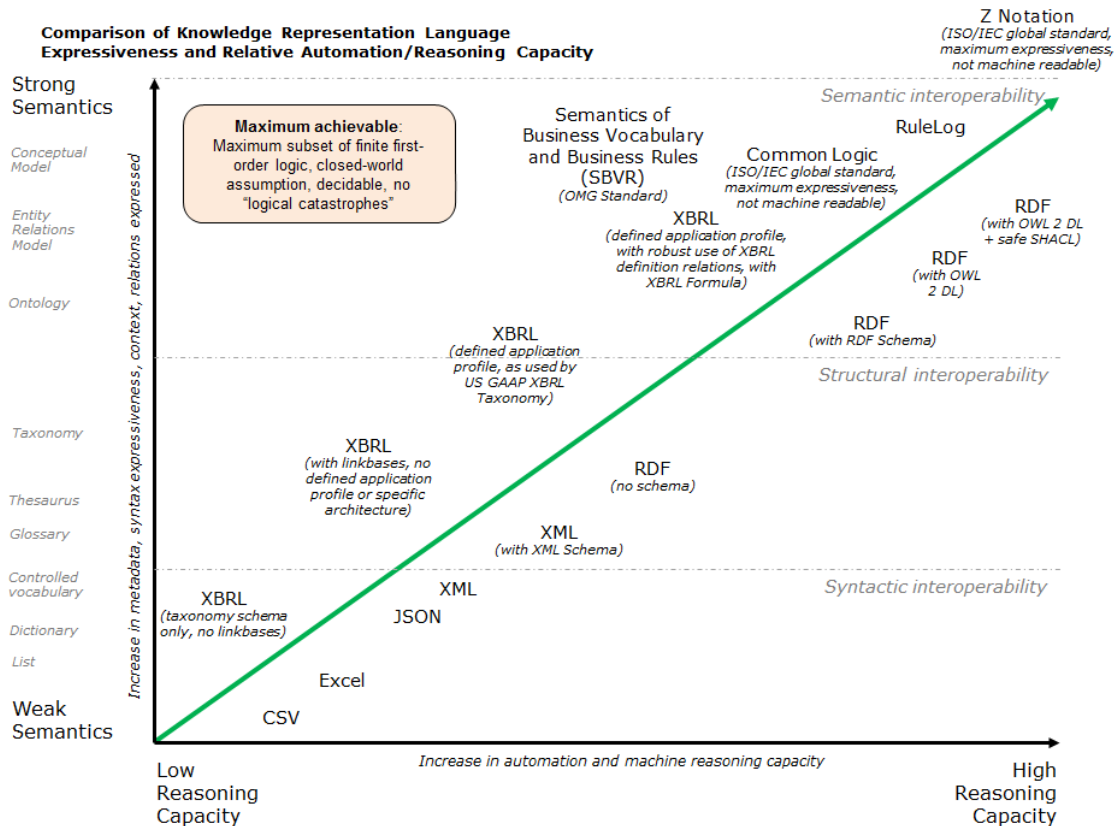
### ***2.3.3. Understanding the relation between expressiveness and reasoning capacity***

Why is the expressiveness of a language important? There are two reasons. First, the more expressive a language the more that language can provide in terms of describing the information being represented and verifying the consistency of what is being represented with expectations (i.e. quality).

But secondly, the more expressive the language is; the more a computer can do for a user of an application in terms of reasoning capacity. The higher the expressiveness of the language, the better the problem solving logic. So, the two work together. Both the quality of the information being processed is higher and what the software can do is higher because of both the expressiveness of the language but also because of the quality of the information which is represented.

Another way to say this is “nonsense in, nonsense out”. As has been pointed out, the only way to have a meaningful exchange of information is the prior existence of technical syntax rules (the language syntax), business domain semantics (the descriptive and structural metadata), and the workflow rules (protocols for what to do if say an amended financial report is submitted to a regulator).

This graphic below compares the relative knowledge representation language expressiveness and the relative automation and reasoning capacity which is achievable using that language.



Inspired by similar comparisons from *An Intrepid Guide to Ontologies* <http://www.mkbereman.com/data/2007/05/16/> and *Semantics Overview* <http://prezi.com/prvskl8p03ln/semantics-overview/>

At the bottom left hand corner of the graphic you see "CSV" which is not expressive (i.e. weak semantics). At the top left you see the ISO/IEC standard "Z Notation" which is highly expressive (i.e. strong semantics). But remember, Z Notation is not machine-readable. But you also see Common Logic, Semantics of Business Vocabulary Rules, RuleLog, SHACL, and XBRL as having strong semantics. Those formats are all machine-readable.

No knowledge representation language is 100% complete. Each has specific, knowable limitations. One must be conscious of such limitations when creating a representation of some problem domain in machine readable form.

A representation language or framework which cannot be measured for simplicity is a recipe for unnecessary complexity. Conscientious knowledge engineers are compelled to express a problem domain's conceptual model as richly as possible. With a highly-expressive language at a knowledge engineer's disposal it is possible to think through different representational options at a level of detail that is impossible with a weaker-expressive language. Stronger languages push one more than one using a weaker language. Testing pushes one more than not using testing toward greater accuracy and comprehensiveness. As is said, "Ignorance is bliss." Limitations of expressiveness of the representation language used should be exposed so that the limitations become conscious.

### 2.3.4. Specific expressiveness features comparison

The following is a comparison of specific features of expressiveness<sup>189</sup>:

## KRR Features Comparison: Rulelog Shines

<b>Feature</b>	<b>System</b>	<b>Rulelog Rules</b> - e.g., Ergo	<b>Datalog Rules</b> - e.g., Jena, SWRL, Ontobroker, SPIN	<b>Production Rules</b> - e.g., IBM, Oracle, Red Hat	<b>Prolog</b> - e.g., SICStus, SWI, XSB	<b>FOL &amp; OWL-DL</b> - e.g., Vampire, Pellet, Prover9	<b>ASP Solvers</b> - e.g., DLV, CLASP
<b>Semantic &amp; on standardization path</b>		✓	✓	restricted case	restricted case	✓	✓
<b>Basic expressiveness</b>							
• Datalog LP		✓	✓	✓	✓	✓	✓
• Logical functions		✓	✗	✗	✓	✓	✓
• General formulas		✓	✗	✗	✗	✓	✗
<b>Full Meta expressiveness</b>							
• Higher-order syntax, provenance		✓	✗	✗	✗	✗	✗
• Defeasibility & well founded negation		✓	✗	✗	✗	✗	✗
• Restraint bounded rationality		✓	✗	✗	✗	✗	✗
• Probabilistic		✓	✗	✗	✗	✗	✗
<b>Efficiency</b>							
• Goal-directed		✓	✗ (except Jena)	✗	✓	✓	✓
• Full LP tabling with dependency-aware updating		✓	✗	✗	✗ (except XSB)	✗	✗
• Polynomial time complexity		✓	✓	✓	✗	✗	✗

Note that it is unknown if any of these knowledge representation logics supports a multidimensional model (i.e. without the user having to create that model). KRR – Knowledge representation and reasoning.

### 2.3.5. Understanding why logical catastrophes break systems

A logical catastrophe is a failure point. Logical catastrophes must be eliminated. Systems should never have these failure points. A basic example of a catastrophic failure is creating metadata that puts a process into an infinite loop that the software will not recover from. This type of catastrophic failure is resolved by simply not allowing the conceptual model to include such structures which cause the possibility of infinite loops. It really is that straight forward.

Here are other types of logical catastrophes:

- **Undecidability:** If a question cannot be resolved to a TRUE or FALSE answer; for example if the computer returns UNKNOWN then unpredictable results can be returned. Logic used by a computer for most business purposes must be decidable. Saying this another way, three-value logic<sup>190</sup>

<sup>189</sup> Coherent Knowledge, *KRR Features Comparison*, <http://coherentknowledge.com/wp-content/uploads/2013/05/talk-main-v14-post.pdf#page=16>

<sup>190</sup> Wikipedia, *Three-valued Logic*, retrieved October 19, 2016, [https://en.wikipedia.org/wiki/Three-valued\\_logic](https://en.wikipedia.org/wiki/Three-valued_logic)

(i.e. TRUE, FALSE, and UNKNOWN are all valid) is a valid for of logic. However, if some people use two value logic (TRUE, FALSE) and others use three-value logic, big problems can occur.

- **Infinite loops:** If a computer somehow enters an infinite loop from which it cannot return because of a logic error or because the logic is too complex for the machine to work with; the machine will simply stop working or return nonsense.
- **Unbounded system structures or pieces:** Systems need boundaries for them to work correctly. Boundaries must be well defined so that they are well understood. If a system does not have the proper boundaries, then a machine can become confused or not understand how to work with information that is provided. For example, if an entirely new class of concept is added to a system that the system has no knowledge of, the system will not understand how to process that class of concept and will fail.
- **Unspecific or imprecise logic:** Confusing precise results with the capabilities of a computer to provide a statistically created result can cause problems. It is not expected that the business system at the level of describing the things in the system be able to support "fuzzy logic" or "probabilistic reasoning" or other such functionality.

### ***2.3.6. Understanding the critical importance of decidability***

There are two fundamental approaches to viewing a system that one could take: the open world assumption (i.e. two-value logic) and the closed world assumption (i.e. three-value logic). Formal logic and relational databases use the closed world assumption. Decidability means that a conclusion can be reached.

- In the **open world assumption** a logical statement cannot be assumed true on the basis of a failure to prove the logical statement. On a World Wide Web scale this is a useful assumption; however a consequence of this is that an inability to reach a conclusion (i.e. not decidable).
- In the **closed world assumption** the opposite stance is taken: a logical statement is true when its negation cannot be proven; a consequence of this is that it is always decidable. In other applications this is the most appropriate approach. Relational databases use this approach.

So each type of system can choose to make the open world assumption or the closed world assumption based on its needs. Because it is important that a conclusion as to the correct mechanics of a financial report is required because consistent and correct mechanics are necessary to making effective use of the information contained within a financial report; the system used to process a financial report must make the closed world assumption.

### ***2.3.7. Inference techniques***

There are various types of inference techniques available<sup>191</sup>.

---

<sup>191</sup> Decision Support Systems and Intelligent Systems, Efraim Turban and Jay E. Aronson 6th ed, Copyright 2001, Prentice Hall, Upper Saddle River, NJ, *Inference Techniques*, Chapter 13, <http://www.indiana.edu/~bnwrbk/K510/ch13.ppt>

### 2.3.8. Setting the right expectation by understanding the capabilities of computers

First-order logic has limitations<sup>192</sup>. Business professionals need to understand these limitations so that they understand what computers can and cannot do, what is hard and what is easy to implement using computers, and to otherwise set their expectations appropriately. Remember, computers cannot perform magic. Computers fundamentally follow the rules of mathematics which follow the rules of formal logic. It really is that straight forward.

It is difficult to get computers to effectively work with information such as the following:

- fuzzy expressions: "It **often** rains in autumn."
- non-monotonicity: "Birds fly, penguin is a bird, but a penguin does not fly."
- propositional attitudes: "Eve **thinks** that 2 is not a prime number." (It is true that she thinks it, but what she thinks is not true.)
- modal logic
  - possibility and necessity: "It is **possible** that it will rain today."
  - epistemic modalities: "Eve **knows** that 2 is a prime number."
  - temporal logic: "I am **always** hungry."
  - deontic logic: "You **must** do this."

While it is possible to implement this sort of functionality within computer systems using technologies such as probabilistic reasoning<sup>193</sup>, those systems will be less reliable and significantly more difficult to create. On the other hand, probabilistic reasoning can provide value. The bottom line is this: what are the boundaries of the system?

### 2.3.9. General versus specific problem solving logics

Problem solving logics can be general or specific. Another term used for general is "weak (basic) problem-solving method". Another term for specific is "strong problem-solving method". Both the general and specific logics have advantages and disadvantages.

General problem solving logics are widely applicable to many problem domains which is an advantage. However, with this flexibility comes the price of a harder to use problem solving logic.

Specific problem solving logics are limited and generally applicable to one specific problem domain. This limitation to one problem domain can be seen as a disadvantage. However, an advantage of the specific nature of the problem solving logic is that it tends to be easier to use because it is specific to the problem domain.

XBRL tends to be limited to be a specific problem solving logic limited to business reporting and financial reporting.

---

<sup>192</sup> Martin Kuba, Institute of Computer Science, *OWL 2 and SWRL Tutorial, Limitations of First-order logic expressiveness*, <http://dior.ics.muni.cz/~makub/owl/>

<sup>193</sup> Wikipedia, *Probabilistic Logic*, retrieved August 28, 2016, [https://en.wikipedia.org/wiki/Probabilistic\\_logic](https://en.wikipedia.org/wiki/Probabilistic_logic)

### 2.3.10. *Approaches for representing information logically in machine readable form*

There are four generally used approaches to representing information logically in machine-readable form:

- **Natural language format:** A natural language which is parsed.
- **Truth table-type format:** A table-type or “truth table” type format.
- **Markup language:** A markup language of some sort.
- **Graphical format:** A graphical format.

### 2.3.11. *Functional layers or categories of problem solving logic*

Problem solving logics can be grouped into “functional layers” or “functional categories” or “functional groups”<sup>194</sup>:

- **Sequence, process or flow:**
  - **procedural logic** – model sequence, loop, or iterative procedures
  - **flow logic** – fully automated sequence of operations, actions, tasks, decisions, rules.
  - **workflow logic** – type of flow logic, semi-automated or manual processes that need an action to be taken from outside the system by another system or human.
- **Information compliance, quality, consistency, completeness, accuracy:**
  - **validation logic:** validate action assertions.
  - **decision logic:** type of validation logic, handles execution que and conflict resolution.
  - **inference logic:** deviations which derives new facts using existing facts, rules, and logical or mathematical reasoning.
  - **structural relations logic:** enforces structural relationships within a representation model

These categories or layers can be useful in grasping the potential power of a problem solving logic.

### 2.3.12. *Details of problem solving logic features*

The comparison below is a DRAFT of a detailing of problem solving logic features that are included in ISO/IEC standard Common Logic. It also tries to articulate the functionality offered by software products to meet the problem solving logic needs when working with a digital financial report.

---

<sup>194</sup> Logic, <http://wiki.flexrule.com/index.php?title=Logic>



Problem Solving Logic Feature	ISO/IEC Common Logic, OMG SBVR, RuleLog	OWL 2 DL, SROIQ, Description Logic	FlexRule	Ergo Suite	XBRL Formula Processor
<b>Basic expressiveness (relational database logic (DATALOG))</b>					
Set theory (cardinality, association, aggregation)					
Constraints and data types					
Referential integrity					
Definition of Terms					
Temporal reasoning					
<b>Assertions</b>					
Structural assertions (including mereology or part-whole)					
Action assertions					
Mathematical assertions (add, subtract, multiply, divide)		using RIF			
<b>Derivations/inference logic (deriving facts)</b>					
Mathematical inference					
Logical inference					
<b>Flow logic (procedural, workflow)</b>					
Procedural					
Workflow					
<b>Chaining</b>					
Forward chaining					
Backward chaining (goal directed)					
<b>Other logical functionality</b>					
Fiscal period reasoning (inherent understanding of)					
Multidimensional model (inherent understanding of)		using DataCube			
Object oriented representations (frames, see CLIPS)					
Logical functions					
<b>Full meta expressiveness</b>					
Higher-order syntax, provenance					
Defeasibility and well-founded negation					
Restraint bounded rationality					
Probabilistic reasoning					

### 2.3.13. Summary of industry initiatives and standards

The following is a summary of industry initiatives and standards that can be used to implement business rules:

- **RuleML:** Rule Markup Language (RuleML) is an international industry initiative that closely collaborates with the W3C on the standardization of high-precision rules. RuleML is a family of closely related designs for different logical languages that overlap. ([http://wiki.ruleml.org/index.php/RuleML\\_Home](http://wiki.ruleml.org/index.php/RuleML_Home); <http://cs.unb.ca/~boley/papers/RuleML-Overarching.pdf> )
- **RIF:** Rules Interchange Format (RIF) is the W3C's design based on RuleML. RIF is a standard for exchanging rules among rule systems. W3C is an umbrella mature standards design organization. (<http://www.w3.org/TR/rif-overview/>)
- **RuleLog:** Rules Logical Programming Language (Rulelog) is an industry initiative to create a logical language designed to be appropriately expressive for supporting knowledge representation in complex domains, such as sciences and law, and yet to be efficiently implementable. (<http://ruleml.org/rif/rulelog/spec/Rulelog.html>)

- **Common Logic:** Common Logic is an ISO/IEC standard based on classical logic. ([https://en.wikipedia.org/wiki/Common\\_Logic](https://en.wikipedia.org/wiki/Common_Logic))
- **SBVR:** Semantics of Business Vocabulary and Business Rules (SBVR) is an OMG standard that was designed and built to be logically equivalent to Common Logic. (<http://www.omg.org/spec/SBVR/>)
- **SWRL:** Semantic Web Rule Language (SWRL) is an industry initiative, a special case of RuleML standards design. (<http://www.w3.org/Submission/SWRL/>)
- **SPIN:** SPARQL Inferencing Notation (SPIN) is an approach to writing rules using SPARQL notation; SPIN is similar to SWRL, a subset of RuleML, and a subset of RuleLog. SPIN is an industry initiative. (<http://www.w3.org/Submission/spin-overview/>)
- **SHACL:** SHACL (Shapes Constraint Language) is a language for validating RDF graphs against a set of conditions. Said another way, it is used for “expressing constraints on RDF data”. And another way, “perform closed-world constraint checks on RDF-based data”. SHACL replaces SPIN. SHACL is a W3C recommendation. (<http://www.w3.org/TR/shacl/>)
- **XBRL:** Extensible Business Reporting Language (XBRL) is an industry initiative standard specialized language for representing business report and financial report information including related rules. (<https://www.xbrl.org/>)

It appears that the “Semantic Web Stack<sup>195</sup>” will end up being RDF, OWL, and SHACL<sup>196</sup>.

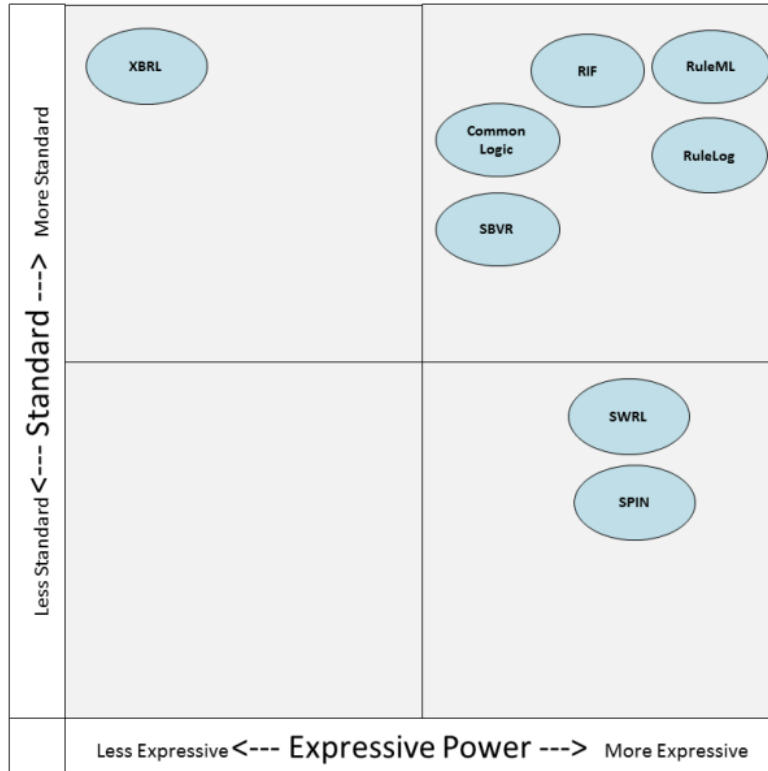
OWL was initially designed by well-intentioned academics. They missed that the open world assumption was a show stopper for business applications. SWRL and SIPN were industry initiatives to resolve issues with OWL. OWL 2 DL added the closed world assumption to OWL. SHACL was created to meet the needs shown by the industry initiatives SWRL and SPIN. SPIN is a subset of SHACL. SHACL achieved the right semantics for closed worlds.

The following graphic shows the relative expressive power and how standard each approach is based on what I have observed and learned:

---

<sup>195</sup> Wikipedia, *Semantic Web Stack*, [https://en.wikipedia.org/wiki/Semantic\\_Web\\_Stack](https://en.wikipedia.org/wiki/Semantic_Web_Stack)

<sup>196</sup> John Walker, <https://twitter.com/wohnialker/status/915982539747028992>



The document *Survey of Knowledge Representations for Rules and Ontologies*<sup>197</sup> created by Benjamin Grosf provides additional details that are helpful in evaluating problem solving logic.

## 2.4. Implementing Problem Solving Logic in Software

You have probably heard that “computers are basically 1s and 0s”. That is true. Implementing a problem solving logic is simply about managing the 1s and 0s.

### 2.4.1. NAND gate

The lowest denominator in implementing logic in software is what is called an NAND gate<sup>198</sup>. All logic systems can be converted into NAND gates. Theoretically, any logic function can be realized by correctly combining together enough NAND gates.

<sup>197</sup> Benjamin Grosf, *Survey of Knowledge Representations for Rules and Ontologies*, <http://coherentknowledge.com/wp-content/uploads/2013/05/Ontolog-Forum-talk-OF-surveyKR-20131024-BNG.pdf>

<sup>198</sup> Wikipedia, *NAND Gate*, retrieved June 7, 2017, [https://en.wikipedia.org/wiki/NAND\\_logic](https://en.wikipedia.org/wiki/NAND_logic)



$$Q = \text{NOT}(A \text{ AND } B)$$

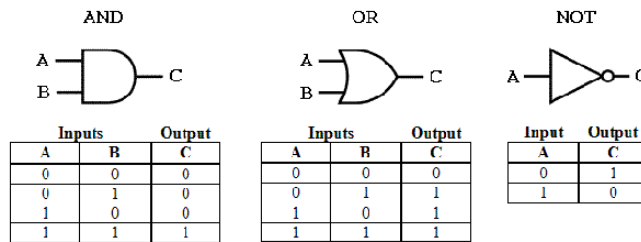
Truth Table

Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0

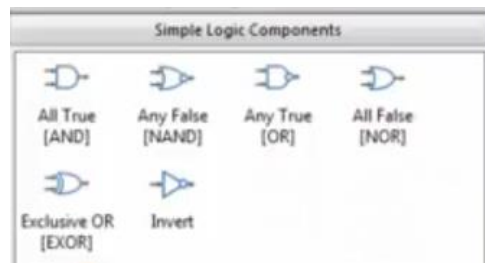
### 2.4.2. Low-level logical functions

Because there are many very common and general combinations of NAND gates that are used over, and over, and over; low-level logical functions are created to make representing that logic easier. Here are common low-level logical functions:

- AND
- NOT
- OR



A function is logically just a combination of NAND gates. Lower level logical components are pieced together to provide the precise logic that you need. Commonly used logical functions are created to make this process easier. Getting software to perform the task that you want is simply piecing together the correct logical building blocks to arrive at the logic that you desire, the software doing precisely what you want the software to do.



### 2.4.3. Higher-level logical functions

To make things even easier to implement, other layers of higher-level logical functions can be implemented. The more specific the set of functions, the easier the functions are to use but the less general the applicability of functions are to general

use cases. However, the more general the function or the lower the level of a function, the harder that function is to use.

Higher-level functions is logically just combinations of lower-level functions which have been combined together.

#### 2.4.4. High-level problem solving logic features

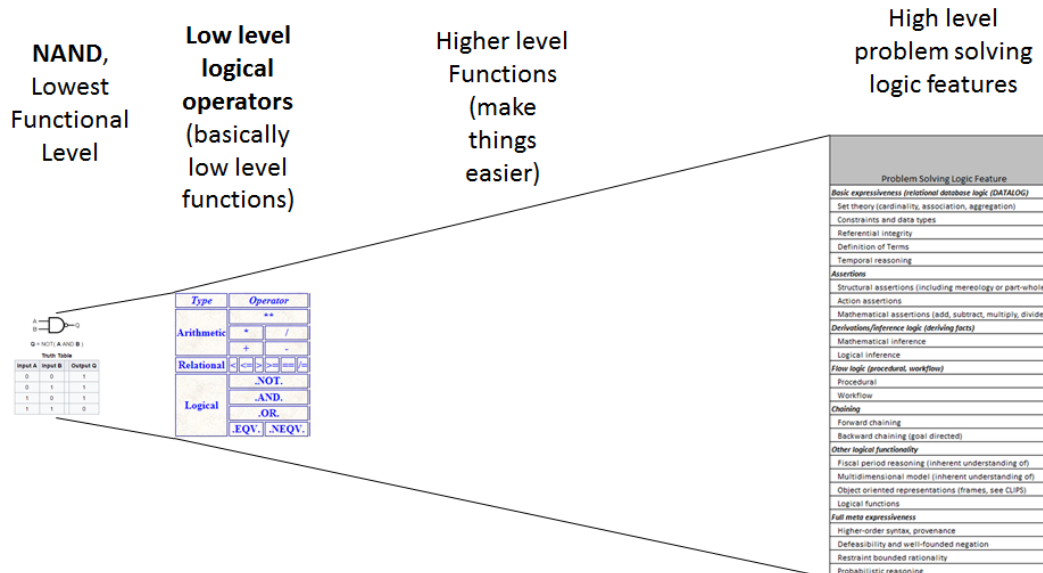
In a previous section, we pointed out high-level problem solving logic features that have general applicability to most problem domains. Problem solving logic features can be grouped into categories:

Problem Solving Logic Feature
<b>Basic expressiveness (relational database logic (DATALOG))</b>
Set theory (cardinality, association, aggregation)
Constraints and data types
Referential integrity
Definition of Terms
Temporal reasoning
<b>Assertions</b>
Structural assertions (including mereology or part-whole)
Action assertions
Mathematical assertions (add, subtract, multiply, divide)
<b>Derivations/inference logic (deriving facts)</b>
Mathematical inference
Logical inference
<b>Flow logic (procedural, workflow)</b>
Procedural
Workflow
<b>Chaining</b>
Forward chaining
Backward chaining (goal directed)
<b>Other logical functionality</b>
Fiscal period reasoning (inherent understanding of)
Multidimensional model (inherent understanding of)
Object oriented representations (frames, see CLIPS)
Logical functions
<b>Full meta expressiveness</b>
Higher-order syntax, provenance
Defeasibility and well-founded negation
Constraint bounded rationality

Problem solving logic features are simply sets of functions that provide specific problem solving logic functionality. Note that problem solving logic must be safely implementable so that the functionality provided is safe, predictable, repeatable, etc.

#### 2.4.5. Summary of problem solving logic capability levels

The diagram below summarizes how lower-level problem solving logic is used to build higher-level problem solving logic features:



### 2.4.6. Implementation alternatives

Business professionals need problem solving logic to solve problems. There are three general areas where the problem solving logic can be implemented:

- Not implemented (i.e. the software user must manually perform the task)
- Implemented in software application (i.e. programmer implementing software creates)
- Implemented in a platform used by the software application (i.e. rules engine)

### 2.4.7. Make or buy decision

Each implementation alternative has pros and cons associated with it. Looking at the basket of pros and cons for each implementation alternative helps one choose the correct implementation alternative.

### 2.4.8. Implementation technology

An automobile is simply metal, plastic, rubber, and glass. The base components of an automobile are always the same; you have an engine, transmission, wheels, seats, body, etc.

However while a Hummer H2 and a Ford Fiesta are both automobiles, a Hummer H2 is a better off-road vehicle. Conversely, a Ford Fiesta is much better on the highway. Arguably, a motorcycle fits into the same set of base components as an automobile; and depending upon how you define "automobile" a motorcycle may or may not be properly categorized with a Hummer H2 and a Fort Fiesta.

You can drive either a Hummer H2, a Ford Fiesta, or a motorcycle off-road or on the highway. The question is, would you really want to do that.

So the question is: what is the best technology to use to implement a problem solving logic. Well, most software vendors would tell you that there product is the best product. Interesting how that tends to work!

You can construct pretty much any set of problem solving logic using any set of tools: RDF/OWL/SHACL and the semantic web stack<sup>199</sup>, business rules management system<sup>200</sup>, Microsoft Visual Basic, relational database management system, Java, MySQL, COBOL, XBRL Formula processor. The real question is: do you REALLY want to do that?

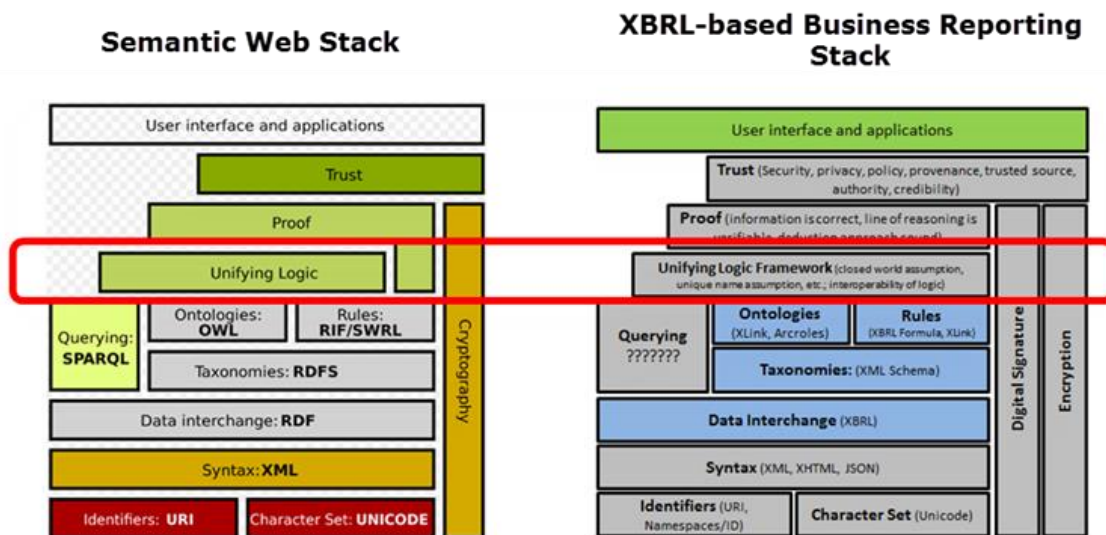
It all boils down to the set of pros and cons associated with each implementation approach and comparing those pros and cons with your needs. The better the match, the more effective and efficient your implementation will be.

## 2.5. Unifying Problem Solving Logic Framework for Business

What if there was one logic framework that everyone used for business applications? So, the logic framework is not the actual business logic. The framework is the logic related tools that one has to use to represent business logic. What if there was a **Unified Problem Solving Logic Framework for Business**<sup>201</sup>?

### 2.5.1. Comparing and contrasting the Semantic Web Stack and XBRL Stack

Below is a comparison of the *Semantic Web Stack* on and a similar diagram of the *XBRL-based Business Reporting Stack*<sup>202</sup>.



Ultimately systems need to interoperate. This is particularly true today in our networked world in our Digital Age. There is an intersection between these two stacks. That intersection is the “Unifying Logic” layer in the Semantic Web Stack and what I named the “Unifying Logic Framework” in the XBRL-based Business Reporting Stack in the diagram.

<sup>199</sup> Wikipedia, *Semantic Web Stack*, retrieved June 7, 2017, [https://en.wikipedia.org/wiki/Semantic\\_Web\\_Stack](https://en.wikipedia.org/wiki/Semantic_Web_Stack)

<sup>200</sup> Wikipedia, *Business Rules Management System*, retrieved June 7, 2017, [https://en.wikipedia.org/wiki/Business\\_rule\\_management\\_system](https://en.wikipedia.org/wiki/Business_rule_management_system)

<sup>201</sup> *Unified Logic Framework for Business*, <http://xbrl.squarespace.com/journal/2017/11/26/unifying-logic-framework-for-business.html>

<sup>202</sup> *Comparing and Contrasting Semantic Web Stack and XBRL Stack*, [http://xbrl.azurewebsites.net/2017/Library/SemanticWebStack\\_XBRLStack.pdf](http://xbrl.azurewebsites.net/2017/Library/SemanticWebStack_XBRLStack.pdf)

The common denominator in different technical implementation by business systems is business logic<sup>203</sup>.

So, for example, the business logic that is referred to as the Accounting Equation<sup>204</sup> is that “Assets = Liabilities + Equity”. That logic is the same in the Semantic Web Stack, or XBRL-based Business Reporting Stack, or any other technical implementation of any technology stack. It will never be the case that if business logic is to work one way in one system that the same business logic would work differently in some other system. Note that this is not the same thing as different systems having different business logic which is, by definition, different.

Another business logic rule with which accountants may not be as directly familiar, but they are indirectly familiar with this rule, notion of *closed world assumption*<sup>205</sup> or as contrast to the *open world assumption*<sup>206</sup>. The reason I say that professional accountants and other business professionals are indirectly familiar with this logic rule is because SQL databases all use the *closed world assumption* in the logic they use for answering questions of their users. The reason this is important to understand is that if one technical architecture was using the *closed world assumption* and another was using the *open world assumption* to answer questions there is a possibility of getting different answers to the exact same question from two different systems, which is an undesirable result.

Finally, any interaction between two systems is limited to the lowest common denominator in terms of the logic framework of the two systems. The reason is that it is only to the degree of logic provided by some logic framework that conveyers of information can represent the business rules that enforce business logic and therefore it is to that degree that quality can be effectively managed. Recall this from our previous discussion about relative expressiveness.

### 2.5.2. Advantages of a unifying problem solving logic framework

The following is a summary of the advantages of having one global standard problem solving logic framework:

- If business professionals interact with systems at the level of business logic, then the business professionals will **find the system approachable** and they will be able to effectively use the system because they are interacting using something they understand, business logic.
- Logic frameworks have different abilities to express business logic. The **MOST POWERFUL** logic framework that is **SAFE TO USE** (i.e. reliable, predictable, repeatable) is what business professionals tend to desire.
- If no one consciously creates some global standard **Unifying Logic Framework for Business**, then the logic frameworks could be incompatible between two business systems that are interacting and business professionals creating, consuming, or otherwise using information using those different systems might not be on par.

---

<sup>203</sup> *Common Denominator is Business Logic*, <http://xbrl.squarespace.com/journal/2017/11/24/common-denominator-is-business-logic.html>

<sup>204</sup> Wikipedia, *Accounting Equation*, [https://en.wikipedia.org/wiki/Accounting\\_equation](https://en.wikipedia.org/wiki/Accounting_equation)

<sup>205</sup> Wikipedia, *Closed World Assumption*, [https://en.wikipedia.org/wiki/Closed-world\\_assumption](https://en.wikipedia.org/wiki/Closed-world_assumption)

<sup>206</sup> Wikipedia, *Open World Assumption*, [https://en.wikipedia.org/wiki/Open-world\\_assumption](https://en.wikipedia.org/wiki/Open-world_assumption)



- **Understanding the boundaries of a logic framework is important;** business professionals need to understand what the logic framework does and does not provide to most effectively employ the logic system.

In my personal view, XBRL International should consider creating such a **Unified Logic Framework for Business** in general, or I guess it could be a **Unified Logic Framework for Business Reporting** specifically. Because the Semantic Web Stack created by the W3C has a significantly more powerful logic framework that has, over the past 25 years been consciously tuned to be completely safe but very, very powerful for business systems and such applications; XBRL is at a significant disadvantage and, in my view, is at risk of becoming less relevant. The most powerful business systems will be those with the most powerful problem solving logic. One standard problem solving logic is better than many different proprietary problem solving logics.

### **2.5.3. Defining a unifying problem solving logic for business**

To define some unifying problem solving logic for business, two alternative approaches exist: (a) use something that exists or (b) create something new. The current alphabet soup of standard logic frameworks for rules, candidates for such a unifying logic framework are perhaps:

- ISO/IEC Common Logic (CL)
- OMG Semantics of Business Vocabulary and Business Rules (SBVR)
- W3C RDFS + OWL + RIF/SWRL syntax logic (SWRL is not a recommendation, only a submission, RIF and SWRL seem to have issues)
- W3C RDFS + OWL + SHACL syntax logic which specifies closed world assumption and unique names assumption
- Industry Initiative RuleLog which is designed to be appropriately expressive for supporting knowledge representation in complex domains and yet to be efficiently implementable
- Industry Initiative RuleML which allows for partially constrained logic profiles and fully-specified logic semantics
- XBRL Formula (which has known deficiencies<sup>207</sup>)

### **2.5.4. Characteristics of such a unifying problem solving logic framework**

The following is a summary of the characteristics of such a unifying problem solving logic framework:

- a global standard logic framework for business
- represented using a controlled natural language format
- represent logic at the highest level possible such that the logic is understandable by business professionals
- rules created are approachable by business professionals

---

<sup>207</sup> *Specific Deficiencies in Capabilities of Existing XBRL Formula Processors*, <http://xbrl.squarespace.com/journal/2016/9/26/specific-deficiencies-in-capabilities-of-existing-xbrl-formu.html>

- built in but optional multidimensional model that does not force the use of OLAP, but usable with OLAP or OLTP
- enables interoperability between technology stacks
- enables interoperability between XBRL, Global Legal Entity Identifier (GLEI)<sup>208</sup>, Financial Industry Business Ontology (FIBO)<sup>209</sup>, Financial Regulation Ontology (FRO)<sup>210</sup>, the US GAAP XBRL Taxonomy, the IFRS XBRL Taxonomy, etc.
- built based on the logic framework of the Semantic Web Stack which has been evolving for 25 or so years.

### 3. Introduction to Expert Systems

This section provides a comprehensive introduction to expert systems that is approachable by professional accountants.

In the *Conceptual Overview of an XBRL-based, Structured Digital Financial Report*<sup>211</sup> I state that digital financial report creation software will be an expert system much like CAD/CAM software which is used to create blueprints. But what is an expert system? How do you create one? What makes expert systems work?

What does that mean? Does it mean that professional accountants will be replaced by machines? If the value that you add is simply clerical, amounting to cutting and pasting information or rekeying information, then quite possibly some of the tasks you perform could be replaced by automated machine-based processes. As explained in the document *Comprehensive Introduction to Knowledge Engineering for Professional Accountants*<sup>212</sup>, there are certain tasks machines can perform and other tasks which machines will never likely be able to perform.

Thousands of expert system tools of all prices and qualities are commercially available today for performing different tasks. Because XBRL-based financial reports are structured data, computer software programs, such as expert systems, offer new capabilities specifically for financial reporting<sup>213</sup>. But what new capabilities will be offered? What work might be automated? This document is intended to help professional accountants understand what an expert system is, how they work, and what capabilities they will bring to help them perform work. Understating or overstating these capabilities are both not helpful.

The point is that setting the right expectations helps one understand what is actually practical and useful. Thinking that technology will have no impact on how you perform your work will be a complete disaster for your career as an accountant and/or for your business.

---

<sup>208</sup> Global Legal Entity Identifier, <https://www.gleif.org/en/about/this-is-gleif>

<sup>209</sup> Financial Industry Business Ontology, <http://www.edmcouncil.org/financialbusiness>

<sup>210</sup> Financial Regulation Ontology, <http://finregont.com/>

<sup>211</sup> *Conceptual Overview of an XBRL-based, Structured Digital Financial Report*, [http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.1\\_ConceptualOverviewOfDigitalFinancialReporting.pdf](http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.1_ConceptualOverviewOfDigitalFinancialReporting.pdf)

<sup>212</sup> *Introduction to Knowledge Engineering for Professional Accountants*, [http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.3\\_KnowledgeEngineeringBasicsForProfessionalAccountants.pdf](http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.3_KnowledgeEngineeringBasicsForProfessionalAccountants.pdf)

<sup>213</sup> YouTube.com, *How XBRL Works*, <https://www.youtube.com/watch?v=nATJBPOiTxM>

The global consultancy firm Gartner classifies XBRL as a transformational technology<sup>214</sup>. Gartner defines transformational as something that "enables new ways of doing business across industries that will result in major shifts in industry dynamics". Major shifts means lots of change and some winners and some losers.

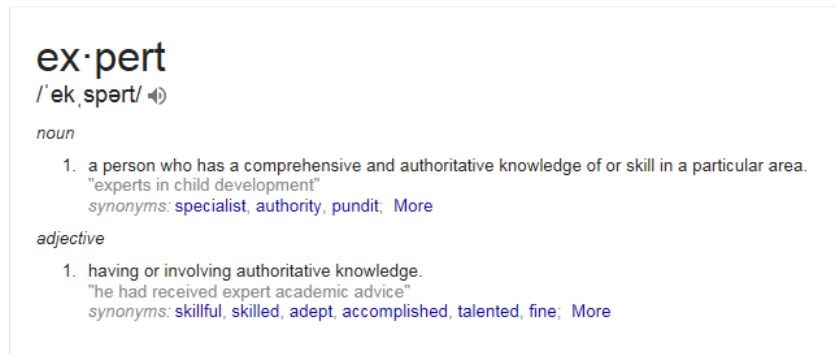
This document helps professional accountants sort through all the information and misinformation that they are hearing related to XBRL-based structured digital financial reporting.

### 3.1. Deconstructing the Notion of an Expert System

To understand expert systems one first needs to understand the notion of an expert. This section is dedicated to setting your perspective as to the notion of an expert and the fundamental notion of an expert system. The section provides specific definitions, deconstructing the pieces so that we can subsequently put the pieces back together.

#### 3.1.1. Definition of an expert

Google defines expert<sup>215</sup> as "a person who has comprehensive and authoritative knowledge of skill in a particular area".



The image shows a screenshot of the Google search results for the word "expert". The word "expert" is displayed in a large, bold font. Below it, the phonetic transcription "/ˈɛkˌspɜrt/" is shown. The word is identified as a "noun" and an "adjective". For the noun, the definition is "a person who has a comprehensive and authoritative knowledge of or skill in a particular area." with an example "experts in child development" and synonyms "specialist, authority, pundit; More". For the adjective, the definition is "having or involving authoritative knowledge." with an example "he had received expert academic advice" and synonyms "skillful, skilled, adept, accomplished, talented, fine; More".

Professional accountants have a comprehensive and authoritative knowledge of and skills in the area of accounting and financial reporting. Accountants are experts at accounting and financial reporting.

#### 3.1.2. Artificial intelligence

Artificial intelligence<sup>216</sup> is a branch of computer science. There are many good descriptions of artificial intelligence<sup>217</sup>. Here is one good definition:

Artificial intelligence is the automation of activities that we associate with human thinking and activities such as decision making, problem solving, learning and so on.

<sup>214</sup> Charles Hoffman and Liv Watson, *XBRL for Dummies*, page 145

<sup>215</sup> Google search, *Expert definition*, retrieved August 21, 2016, <https://www.google.com/search?q=expert+definition>

<sup>216</sup> *Introduction to Artificial Intelligence Terminology*, <http://xbri.squarespace.com/journal/2016/7/21/introduction-to-artificial-intelligence-terminology.html>

<sup>217</sup> AlanTuring.net, *What is Artificial Intelligence?*, [http://www.alanturing.net/turing\\_archive/pages/reference%20articles/What%20is%20AI.html](http://www.alanturing.net/turing_archive/pages/reference%20articles/What%20is%20AI.html)

Another more neutral term that basically means the same thing is “machine intelligence”<sup>218</sup>. Think of these tools as narrowly focused employees with great memories that are very good at performing one specific repetitive task well, over, and over, and over. Literally, like a machine.

Those trying to make artificial intelligence work over the past 40 or so years have had limited success<sup>219</sup>. But that is changing. People are putting the pieces together and the technology created from AI research are now available to experiment with. If expectations are not set too high, very useful functionality for limited, narrow problems can be successfully created. Both under estimating or over estimating the capabilities the computer software will be able to achieve can have catastrophic consequences.

One good example of using artificial intelligence is driverless cars. Many people get confused as to what is truly achievable and practical when it comes to driverless cars. Driverless cars are on the streets of Singapore today<sup>220</sup>. While still in prototype mode now to work out details, by 2018 these taxis are anticipated to be commercially available. Uber is testing autonomous cars in Pittsburg<sup>221</sup>.

One of the best ways to understand the capabilities of artificial intelligence is to try it out. Go test drive a Tesla which has driver-assist features.

Artificial intelligence programs that provide expert-level proficiency in solving problems by bringing to bear a body of knowledge about specific tasks are called knowledge based systems or expert systems.

There are two types of artificial intelligence, specialized and generalized:

- **Specialized:** An example of specialized artificial intelligence is programming a computer to play chess. The software performs one specific task. Specialized artificial intelligence is fairly easy to achieve.
- **Generalized:** An example of generalized artificial intelligence is the sort of science-fiction stuff you see in the movies. Why do you only see this in the movies? Because generalized artificial intelligence is extremely hard to make work.

### 3.1.3. Expert systems

Expert systems<sup>222</sup> is a branch of artificial intelligence. Expert systems, also called knowledge based systems or simply knowledge systems, are computer programs. The following is a definition of an expert system:

Expert systems are computer programs that are built to mimic human behavior and knowledge. Expert systems are for reconstructing the expertise and reasoning capabilities of qualified experts within some limited, narrow

---

<sup>218</sup> Shivon Zilis and James Cham, Harvard Business Review, *The Competitive Landscape for Machine Intelligence*, <https://hbr.org/2016/11/the-competitive-landscape-for-machine-intelligence>

<sup>219</sup> John F. Sowa, *Why Has AI Failed? And How Can it Succeed?*, <http://www.ifsowa.com/pubs/micai.pdf>

<sup>220</sup> Reuters, *First driverless taxis hit the streets of Singapore*, <http://www.reuters.com/article/us-autos-selfdriving-singapore-idUSKCN1100ZG>

<sup>221</sup> The Atlantic, *Anybody Can Test a Driverless Car in Pennsylvania*, <http://www.theatlantic.com/technology/archive/2016/09/anybody-can-test-a-self-driving-car-in-pennsylvania/499667/>

<sup>222</sup> *Understanding the Components of an Expert System*, <http://xbri.squarespace.com/journal/2016/5/24/understanding-the-components-of-an-expert-system.html>

domain of knowledge in machine-readable form. A model of the expertise of a domain of knowledge of the best practitioners or experts is formally represented in machine-readable form and the expert system reaches conclusions or takes actions based on that information when trying to solve some problem. The computer program performs tasks that would otherwise be performed by a human expert.

Expert systems are the most commercially successful applications of artificial intelligence research<sup>223</sup>. There are currently thousands of expert systems employed world-wide in industry and government.

Key to understating the capabilities of expert systems is an understanding the fundamental capabilities of computers. To understand the capabilities of computers, please be sure to read the document *Comprehensive Introduction to Knowledge Engineering for Professional Accountants*<sup>224</sup>.

### **3.1.4. Differentiating the mechanical aspects of a financial report and judgment**

Computers are dumb beasts. Computers only follow instructions. Computers will never possess judgment. They may try and mimic judgment and could be successful to some degree.

A financial report itself is mechanical. While what must go into the financial report requires the judgement of a professional accountant, the mechanics of a financial report are objective. Balance sheets always balance<sup>225</sup>. Roll ups always roll up. Roll forwards always roll forward. Everything should always “tick and tie”, “cross cast and foot”. Accountants excel at performing these detailed tasks. But computers are completely capable of managing the mechanical details of a financial report. That frees professional accountants from having to worry about those mechanical details and focus on where they add the most value which is the aspects of creating a financial report that require judgment.

Not making the proper distinction between the mechanical aspects and the aspects that require judgement will cause someone to either overestimate the work a computer can perform or under estimate that work.

### **3.1.5. Business rules**

Key to employing artificial intelligence and therefore making an expert system work is machine-readable business rules<sup>226</sup> of a domain of knowledge.

Business rules guide, control, suggest, or influence behavior. Business 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. Business rules help shape judgment, help make decisions, help evaluate, help shape behavior, and help reach conclusions.

---

<sup>223</sup> Edward Feigenbaum et. al, *KNOWLEDGE-BASED SYSTEMS IN JAPAN*, <http://www.wtec.org/loyola/kb/execsum.htm>

<sup>224</sup> *Introduction to Knowledge Engineering for Professional Accountants*, [http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.3\\_KnowledgeEngineeringBasicsForProfessionalAccountants.pdf](http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.3_KnowledgeEngineeringBasicsForProfessionalAccountants.pdf)

<sup>225</sup> Wikipedia, *Accounting Equation*, [https://en.wikipedia.org/wiki/Accounting\\_equation](https://en.wikipedia.org/wiki/Accounting_equation)

<sup>226</sup> *Comprehensive Introduction to Business Rules*, [http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.4\\_ComprehensiveIntroductionToBusinessRules.pdf](http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.4_ComprehensiveIntroductionToBusinessRules.pdf)

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

### **3.1.6. Objects as rules**

Besides business rules, expert systems can use another form of knowledge representation. Objects, rather than rules, can play a leading role in knowledge representation. Although, object-based representations of knowledge can be regarded as rules organized in a different manner.

### **3.1.7. Human-readable and machine-readable business rules**

Business professionals interact with business rules every day by may not even realize it. Most business rules are in human readable form. But business rules can be represented in both human-readable form and machine-readable form. With the move to digital, more and more business rules are being represented in both human readable form and more importantly machine-readable form. Formalized, standardized machine-readable business rules can help automate processes which have been manual in the past.

### **3.1.8. Intelligent software agents**

As stated, artificial intelligence is the automation of activities that we associate with human thinking and activities such as decision making, problem solving, learning and so on. How do you create that sort of automation? The answer is using a style of computer programming called intelligent software agents.

An intelligent software agent<sup>227</sup> is software that assists people and acts on their behalf. Intelligent agents add value by allowing people to:

- delegate work that they could have done to the agent software.
- perform repetitive tasks,
- remember things you forgot,
- intelligently find, filter and summarize complex information,
- customize information to your preferences,
- learn from you and even make recommendations to you.

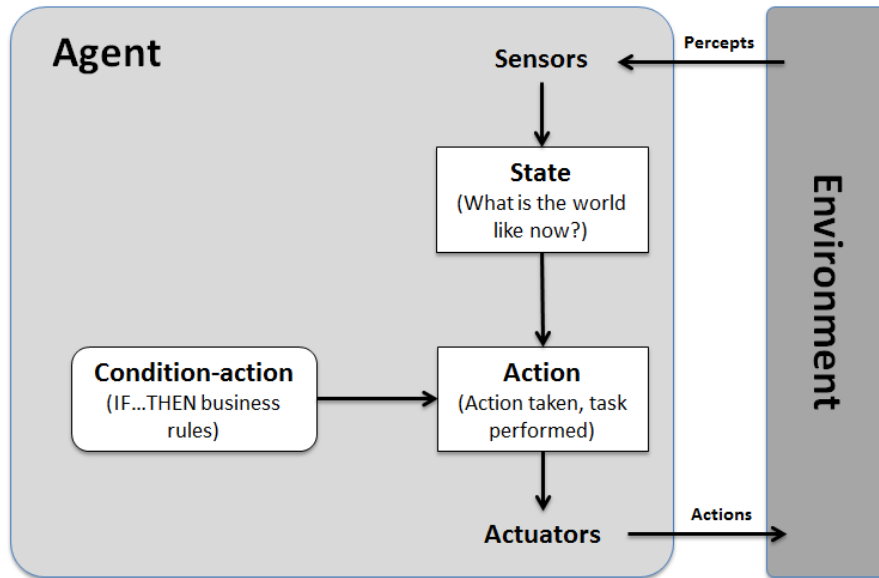
An intelligent agent is computer software capable of **sensing** the **state** of its **environment** and **acting** upon it based on a set of specified **rules**. An intelligent agent performs specific tasks on behalf of another. In the case of software, an agent is a software program. There are many different types of intelligent software agents<sup>228</sup>.

---

<sup>227</sup> *Comprehensive Introduction to Intelligent Software Agents for Professional Accountants*, <http://xbrlsite.azurewebsites.net/2016/Library/ComprehensiveIntroductionToIntelligentSoftwareAgentsForProfessionalAccountants.pdf>

<sup>228</sup> Wikipedia, *Intelligent Agent*, Retrieved July 24, 2016; [https://en.wikipedia.org/wiki/Intelligent\\_agent](https://en.wikipedia.org/wiki/Intelligent_agent)

# Simple Reflex Agent



The document *Comprehensive Introduction to Intelligent Software Agents for Professional Accountants*<sup>229</sup> goes into significantly more detail on the topic of intelligent software agents.

## 3.2. Digging Deeper into Expert Systems and Knowledge Based Systems

The previous section sets a foundation for understanding expert systems. In this section we go into additional important details that help round out your understanding of expert systems and other knowledge based systems.

Simply put, a **knowledge based system** is a system that draws upon the knowledge of human experts 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** to solve problems that normally would require human effort and thought to solve. The knowledge based system supplies an **explanation and justification mechanism** to help system users to understand the **line of reasoning** used and support **conclusions reached** by the knowledge based system and presents that information to the user of the system.

An expert system is a type of knowledge based system.

Humans augmented by machine capabilities, much like an electronic calculator enabling a human to do math quicker, will empower knowledge workers who know how to leverage the use of those machines.

<sup>229</sup> *Comprehensive Introduction to Intelligent Software Agents for Professional Accountants*, <http://xbrl.azurewebsites.net/2016/Library/ComprehensiveIntroductionToIntelligentSoftwareAgentsForProfessionalAccountants.pdf>

### 3.2.1. *Creating an expert system or knowledge based system*

Creating a knowledge based system involves the transformation of machine-readable instructions in such a way as to explain to a machine how a system works and how to make a system work the way you want that system to work.

Then, brick-by-brick, much like building a house, business domain experts and software engineers can create tools that automate certain types of tasks in that process. Humans encode information, represent knowledge, and share meaning using machine-readable patterns, languages, and logic.

That will be the way an increasing number of work tasks will be performed in the Digital Age of accounting, reporting, and auditing. The result will be more efficient processes.

### 3.2.2. *Types of expert systems*

Frank Puppe explains in his book *Systematic Introduction to Expert Systems*<sup>230</sup> 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. The system tends to be instructional in nature.
- **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 or create predictions based on the system.

The assembly of a financial report can be assisted by a construction-type expert system. Helping professional accountants understand what goes into that financial report can be assisted by a classification-type expert system. Creating forecasts and projections of future financial reports can be assisted by simulation-type expert systems.

### 3.2.3. *Components of an expert system*

A software based expert system has four primary components<sup>231</sup>:

- **Database of facts:** A database of facts is a set of observations about some current situation or instance. The database of facts is "flexible" in that they apply to the current situation. The database of facts is machine-readable. An example of a database of facts is information reported a financial report. An XBRL instance is a database of facts.
- **Knowledge base of rules:** A knowledge base is a set of universally applicable rules created based on experience and knowledge of the practices of the best domain experts generally articulated in the form of IF...THEN statements or a form that can be converted to IF...THEN form. At the highest level the knowledge base contains a conceptual model, definitions of things that make up that model, and relationships between the things in the model (types of things, structure of things, parts of things, mathematical relations

---

<sup>230</sup> Frank Puppe, *Systematic Introduction to Expert Systems, Knowledge Representations and Problem-Solving Methods*, page 11 (Note that you can read Parts I and II on Google Books here, <https://books.google.com/books?id=kKqCAAQBAJ>)

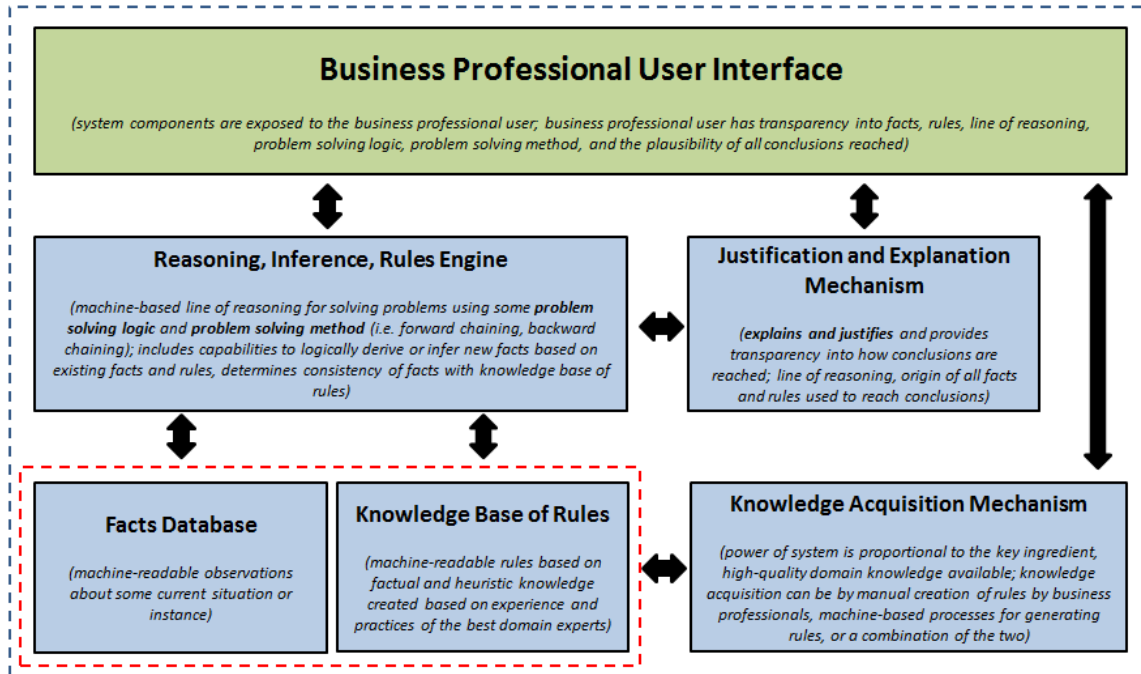
<sup>231</sup> Edward Feigenbaum Chair, et. al., *KNOWLEDGE-BASED SYSTEMS IN JAPAN*, [http://www.wtec.org/loyola/kb/c3\\_s2.htm](http://www.wtec.org/loyola/kb/c3_s2.htm)



between things, etc.). A knowledge base is "fixed" in that its rules are universally relevant to all situations covered by the knowledge base. Not all rules are relevant to every situation. But where a rule is applicable it is universally applicable. All knowledge base information is machine-readable. Business rules are declarative in order to maximize use of the rules and make it easy to maintain business rules. Knowledge that makes up the knowledge base is acquired using manual or automated knowledge acquisition processes. An XBRL taxonomy is a knowledge base of rules.

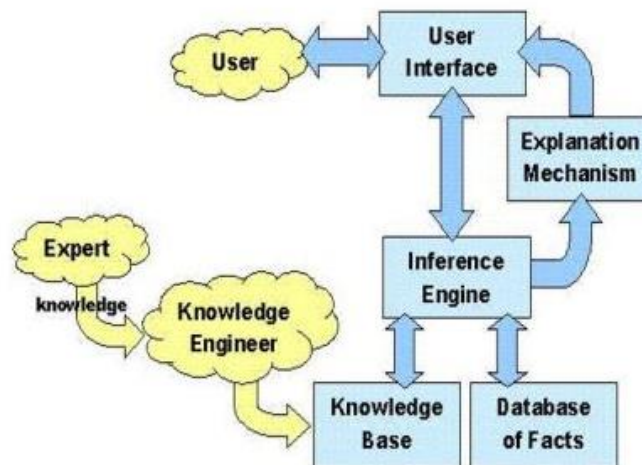
- **Reasoning/inference/rules engine:** A reasoning engine provides a machine-based line of reasoning for solving problems. The reasoning engine processes facts in the fact database, rules in the knowledge base. A reasoning engine is also an inference engine and takes existing information in the knowledge base and the database of facts and uses that information to reach conclusions or take actions. The inference engine derives new facts from existing facts using the rules of logic. The reasoning engine is a machine that processes the information. A reasoning engine has a specific problem solving logic and uses some problem solving method. An XBRL Formula processor, if built correctly, can be a reasoning engine and can perform logical inference.
- **Justification and explanation mechanism:** When an answer to a problem is questionable, we tend to want to know the rationale behind the answer. If the rationale seems plausible, we tend to believe the answer. The justification and explanation mechanism explains and justifies how a conclusion or conclusions are reached. It walks you through which facts and which rules were used and the line of reasoning used to reach a conclusion. The explanation mechanism is the results of processing the information using the rules processor/inference engine and justifies why the conclusion was reached. The explanation mechanism provides both provenance and transparency to the user of the expert system so that the user of the system understands the origin of all facts and rules.
- **Knowledge acquisition mechanism:** The power of a knowledge base system is proportional to the key ingredient of that system: high-quality domain knowledge available in machine-readable form. Knowledge acquisition can be by manual creation of rules by business professionals, machine-based processes for generating rules, or a combination of the two.

These pieces are exposed to the users of the expert system within software applications. One philosophical difference that knowledge based systems tend to have from typical software systems which tend to be procedural in nature is that knowledge based systems separate business domain logic and program control logic. This enables business domain logic to be managed/maintained by business domain experts and that the business domain logic is reusable by other software programs. Below is a summary of the components of an expert system or knowledge based system:



Expert systems provide *transparency* to their users and can explain the solutions they provide by quoting the knowledge used to reach that solution. Single pieces of knowledge can be easily added, changed, or removed; providing *flexibility*. The users of expert systems should require no knowledge of programming languages by either the creator of the expert system or user of the expert system, providing *ease of use*. The boundaries of the problem solving capabilities should be clear so users of the system understand what the system provides and what needs to be provided using alternative processes.

The following is another graphic<sup>232</sup> that shows how the components of an expert system interact with one another:



<sup>232</sup> IMS MBA, *Architecture of an Expert System*, <https://imscdrmba.wordpress.com/206-unit-iii/> ; (direct line to image) <https://imscdrmba.files.wordpress.com/2016/04/expert-systems.jpg>

Software applications must provide all the pieces of the system. The *law of irreducible complexity* states basically that "A single system which is composed of several interacting parts that contribute to the basic function, and where the removal of any one of the parts causes the system to effectively cease functioning." That means that each of the parts need to exist for the system of an XBRL-based digital financial report to work correctly.

Further, the system must be usable by business professionals. The *law of conservation of complexity* essentially states, "Every software application has an inherent amount of irreducible complexity. That complexity cannot be removed from the software application. However, complexity can be moved. The question is: Who will have to deal with the complexity? Will it be the application user, the application developer, or the platform developer which the application leverages?"

### **3.2.4. Knowledge acquisition**

A key ingredient in a knowledge based system is domain knowledge. The power of any knowledge based system is proportional to the high-quality domain knowledge available within that system. The fact that a thick metadata layer and the benefits of that metadata in terms of getting a computer to be able to perform useful and meaningful work is not disputed. What is sometimes disputed is *how* to most effectively and efficiently get that thick metadata layer. There are two basic approaches to getting this metadata, or machine-readable business rules, that makes up the knowledge base:

- **Have the computer figure out what the metadata is:** This approach uses artificial intelligence, machine learning, and other high-tech approaches to detecting patterns and figuring out the metadata.
- **Tell the computer what the metadata is:** This approach leverages business domain experts and knowledge engineers to piece together the metadata manually so that the metadata becomes available.

Because knowledge acquisition can be slow and tedious, much of the future of knowledge based systems depends on breaking the knowledge acquisition bottleneck and in codifying and representing a large knowledge infrastructure. However, this is not an "either/or" question. Both manual and automated knowledge acquisition methods can be used together.

### **3.2.5. Problem solving method**

The objective of an expert system is to solve some problem. Conventional software applications work using sequential algorithms (software programs) and data. Expert systems separate the "algorithms" into two parts; knowledge and the problem solving method. Another way to say this is that expert systems separate software programs into domain logic and control logic.

- Forward chaining
- Backward chaining
- Combination of forward and backward chaining

Please see the document *Comprehensive Introduction to Business Rules for Professional Accountants*<sup>233</sup> for a more complete introduction to forward and backward chaining.

### 3.2.6. Problem solving logic

One of the most complicated things to understand about expert systems is the problem solving logic used by the expert system to solve problems. The problem solving logic drives the extent of rules that can be created because the nature of the logic constrains what rules can be expressed.

A logic can be defined as any precise notation for expressing statements that can be judged to be either true or false<sup>234</sup>. Almost any declarative notation, graphic or linear, could be treated as a version of logic: just specify it precisely. A logic needs to define foundational terminology such as: *there exists, every, and, or, if and only if, if-then, not, true, and false*<sup>235</sup>. Really, it is that simple. What makes this complicated are all the different notations that are used to refer to those terms. Using natural language make logic more understandable to business professionals.

Determining the problem solving logic is a balancing act. The objective is to have the maximum amount of expressiveness but the minimum chance that software will break. The logic needs to be safely implementable by software.

For the past 30 or so years, many different technical solutions have been created to solve different business problems. Few of these technical solutions achieved the appropriate balance or equilibrium and tended to not maximize expressiveness or not be safely implementable in the form of software applications. Benjamin Grosf, Michael Kifer, and Mike Dean summarize this history in their presentation, *Semantic Web Rules: Fundamentals, Applications, and Standards*<sup>236</sup>.

Please see the document *Comprehensive Introduction to Knowledge Engineering for Professional Accountants*<sup>237</sup> for a complete discussion of this topic.

The bottom line is that the best balance between expressiveness and safe implementation has been achieved by the ISO/IEC global standard Common Logic. **Common Logic**<sup>238</sup> is a framework for a family of logic languages, based on first-order logic, intended to facilitate the exchange and transmission of knowledge in computer-based systems. That safely expressive sweet spot is also used by the OMG standard **Semantics of Business Vocabulary and Business Rules**<sup>239</sup> which was consciously designed to be logically equivalent to ISO/IEC Common Logic.

---

<sup>233</sup> *Comprehensive Introduction to Business Rules*, [http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.4\\_ComprehensiveIntroductionToBusinessRules.pdf](http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.4_ComprehensiveIntroductionToBusinessRules.pdf)

<sup>234</sup> John F. Sowa, *Fads and Fallacies about Logic*, page 2, <http://www.jfsowa.com/pubs/fflogic.pdf>

<sup>235</sup> Wikipedia, *List of Logical Symbols*, [https://en.wikipedia.org/wiki/List\\_of\\_logic\\_symbols](https://en.wikipedia.org/wiki/List_of_logic_symbols)

<sup>236</sup> Benjamin Grosf, Michael Kifer, and Mike Dean, *Semantic Web Rules: Fundamentals, Applications, and Standards*; <http://coherentknowledge.com/wp-content/uploads/2013/05/talk-prelim-aaai13-rules-tutorial.pdf>

<sup>237</sup> *Introduction to Knowledge Engineering for Professional Accountants*, [http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.3\\_KnowledgeEngineeringBasicsForProfessionalAccountants.pdf](http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.3_KnowledgeEngineeringBasicsForProfessionalAccountants.pdf)

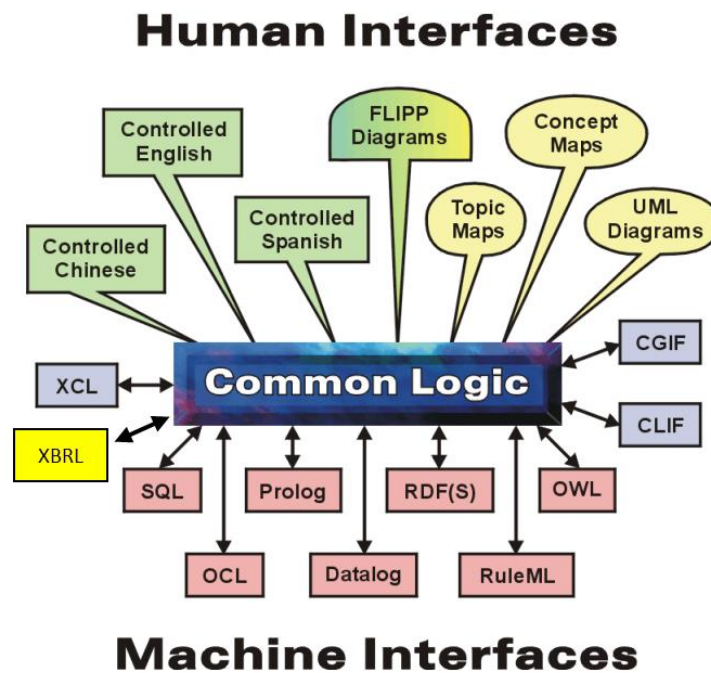
<sup>238</sup> *Understanding Common Logic*, <http://xbrl.squarespace.com/journal/2016/6/23/understanding-common-logic.html>

<sup>239</sup> OMG, *Semantics of Business Vocabulary and Business Rules (SBVR)*, section 2.5 Conformance of an SBVR Processor, page 7, <http://www.omg.org/spec/SBVR/1.0/>

**Rulelog**<sup>240</sup> is a logic for expressing knowledge that is consciously engineered to be consistent with ISO/IEC Common Logic and OMG Semantics of Business Vocabulary and Business Rules. Rulelog is a dialect of W3C's RIF<sup>241</sup>. RuleML<sup>242</sup> is a syntax for implementing rules. Other standard and proprietary syntaxes exist for implementing rules.

The most important thing to realize is that there is a good, safe target in terms of an expressive logic that is also safely implementable in software so catastrophic failures are avoided. Another very good thing is that business professionals don't need to understand the underlying technical details of these logic standards, nor will they ever have to deal with them. Higher level languages that follow the foundations set by Common Logic, Semantics of Business Vocabulary and Business Rules, and Rulelog.

The following graphic shows the role Common Logic<sup>243</sup> plays, establishing a family of logical dialects shared between different software syntax implementations: (note that this graphic was modified, XBRL was added)



The W3C recommended standard syntax is RDF/OWL/SHACL, the semantic web stack. The language in which a problem is stated has no effect on complexity. Reducing the expressive power of a logic does not solve any problems faster; its only effect is to make some problems impossible to state<sup>244</sup>.

<sup>240</sup> Rulelog, <http://ruleml.org/rif/rulelog/spec/Rulelog.html>

<sup>241</sup> W3C, RIF Overview (Second Addition), <http://www.w3.org/TR/rif-overview/>

<sup>242</sup> RuleML, [http://wiki.ruleml.org/index.php/RuleML\\_Home](http://wiki.ruleml.org/index.php/RuleML_Home)

<sup>243</sup> John F. Sowa, *Common Logic: A Framework for a Family Of Logic-Based Languages*, page 5, <http://www.ifsowa.com/ikl/SowaST08.pdf>

<sup>244</sup> John F. Sowa, *Fads and Fallacies about Logic*, page 5, <http://www.ifsowa.com/pubs/fflogic.pdf>

### 3.2.7. Inference techniques

There are various types of inference techniques available<sup>245</sup>. See the problem solving logic document.

### 3.2.8. Benefits of an expert system

Benefits from the use of expert systems include:

- **Reduced costs by using automation:** elimination of routine, boring, repetitive, mundane, mechanical, rote tasks that can be automated
- **Increased uniformity:** consistent answers from the same question or facts; computers are good at performing repetitive, mechanical tasks whereas humans are not; computers do not make mistakes and are good at repeating exactly the same thing each time; performance level is consistent
- **Reduced down-time:** computer based expert systems are tireless and do not get distracted
- **Increased availability:** computer based expert systems are always available simultaneously in multiple places at one time; you get quick response times and can replace absent or scarce experts; convenient
- **Diligence and tenacity:** computers excel at paying attention to detail; they never get bored or overwhelmed and they are always available and will keep doing their job until the task is complete with the same attention to detail
- **Basis for training:** the best practices of the best practitioners can be available to those that are new to and learning about a domain of knowledge
- **Longevity and persistence:** computer based expert systems do not change jobs or retire so knowledge gathered by an organization can remain within that organization
- **Productivity:** computer based expert systems are cheaper than hiring experts and costs can be reduced at the same time that quality increases resulting in increased productivity
- **Multiple opinions:** Systems can integrate the view of multiple experts within a system and choose between the preferred view of multiple expert opinions in the same system
- **Objectivity:** computers apply the same inductive and deductive logic consistently; emotion and personal preferences can be eliminated where they should be eliminated; expert systems do not discriminate
- **Easier dissemination of knowledge:** expert systems are software and metadata and therefore once you have that software and metadata reproducing another version is trivial and the incremental cost is extremely low

In a knowledge based or expert system; knowledge is explicitly represented and can be evaluated, knowledge is permanent, knowledge is easily replicated, and the system is consistent. Operating costs of an expert system are low. Financial report

---

<sup>245</sup> Decision Support Systems and Intelligent Systems, Efraim Turban and Jay E. Aronson 6th ed, Copyright 2001, Prentice Hall, Upper Saddle River, NJ, *Inference Techniques*, Chapter 13, <http://www.indiana.edu/~bnwrbk/K510/ch13.ppt>

creation software of the future will be an expert system which operates similar to how CAD/CAM software for creating blueprints.

### 3.2.9. Disadvantages of expert systems

Everything has advantages and disadvantages. The following can be disadvantages of expert systems:

- **Initial cost:** the initial cost of creating an expert system can be high; the primary cost is for creation of the expert knowledge which is used by the system
- **Maintaining knowledge:** human experts constantly update their knowledge through interaction with other experts, new ideas, common sense, etc.; expert systems have to be maintained to keep knowledge current
- **Garbage in, garbage out:** an expert system is only as good as the machine-readable knowledge which the system uses
- **No common sense:** humans have common sense, expert systems do not
- **Lacks human touch:** expert systems are computer application and have the same characteristics of a computer; they have no compassion, no intuition, cannot exercise real judgment, etc.
- **Inflexibility:** a system, once set up, is inflexible or rather only flexible to the extent that new knowledge is added to the system
- **Restricted:** an expert system usually has expertise in one specific domain of knowledge and is therefore restricted to that specific knowledge

To make expert systems work effectively, disadvantages must be overcome and expert systems should be used to solve problems they are truly capable of solving. Setting the right expectations is important.

### 3.2.10. Contrasting universal tools and domain specific tools

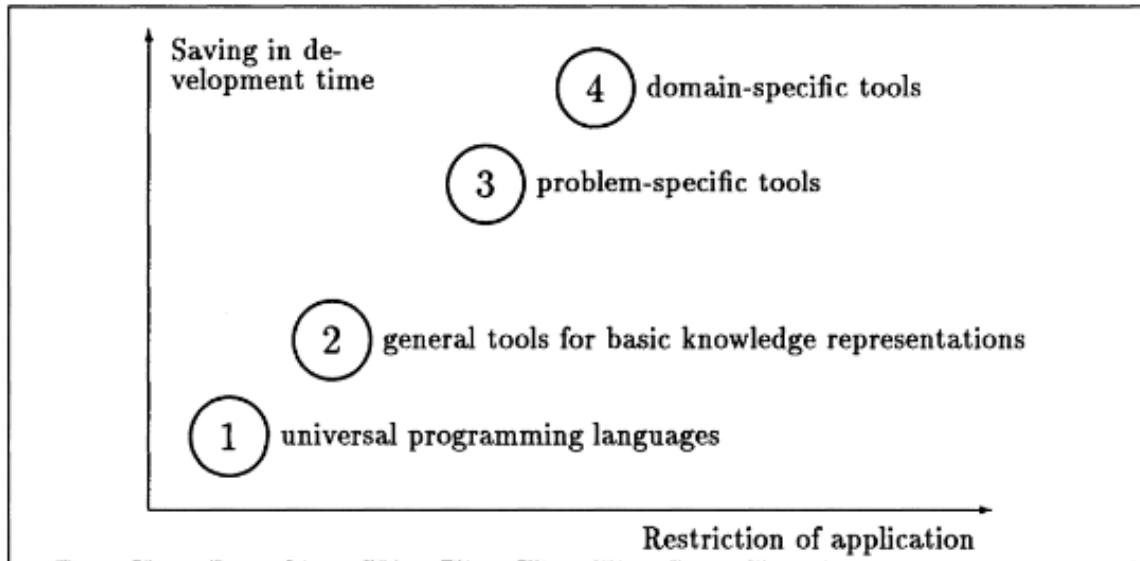
In his book *Systematic Introduction to Expert Systems*<sup>246</sup>, Frank Puppe provides the graphic below. The graphic basically points out that universal, general tools are less restrictive but cost more to create domain-specific tools. In addition to universal, general tools being more costly to create and more difficult to create; domain specific tools are easier to create and much, much easier for business professionals to use because of the restrictions.

So, a “restriction” is not a flaw. The restriction is what makes the tool easier to use and cost less and make easier to develop. You don’t need the universe of all possible options; you only need to create what that specific domain needs. As long as you get these restrictions correct, they really are not “restrictions” of the domain, they are the “boundaries” of the domain. You don’t need them. Technical people don’t typically understand these domain boundaries. Many times to play it safe technical people add flexibility in order to make certain that business domain user needs are being met. But this flexibility comes at a cost. Additional costs are incurred to create the flexibility and software is harder to use because business professionals need to figure out which option they should use.

---

<sup>246</sup> Frank Puppe, *Systematic Introduction to Expert Systems*, page 11, [https://books.google.com/books?id=kKqCAAQBAJ&printsec=frontcover&source=gbs\\_ge\\_summary\\_r&ad=0#v=onepage&q&f=false](https://books.google.com/books?id=kKqCAAQBAJ&printsec=frontcover&source=gbs_ge_summary_r&ad=0#v=onepage&q&f=false)

Business domain people do understand the boundaries if they think about them. Many business professionals cannot properly articulate the appropriate boundaries or restrictions. This communications problem tends to lead to software that costs more to create than is necessary and harder to use than necessary.



This is not an either-or choice. Sometimes universal tools are very appropriate. Other times domain-specific tools are appropriate. Being conscious of these dynamics will lead to the right software being created and the appropriate level of usability. Universal tools are not a panacea. Unconsciously constricting a domain-specific tool when it would have been better to create a more universally usable tool also can be a mistake one makes.

### 3.2.11. Examples of expert systems in other domains

Expert systems are available commercially at different price levels and with different capabilities. The following is a brief list of expert systems to give you an idea of the potential capabilities of expert systems:

- Chess game (for example, IBM's Deep Blue beat the grand master at the time)
- Medical triage and diagnosis
- Robotic surgery
- Aircraft accident investigation
- Patriot missile guidance system
- Numerically controlled manufacturing machine



### 3.3. Using CLIPS to Understand Expert Systems

CLIPS<sup>247</sup> is a tool for building expert systems originally developed by NASA. Since it was first released in 1986 it has undergone significant enhancements and was put into the public domain by NASA in about 2002. CLIPS continues to be maintained as public domain software.

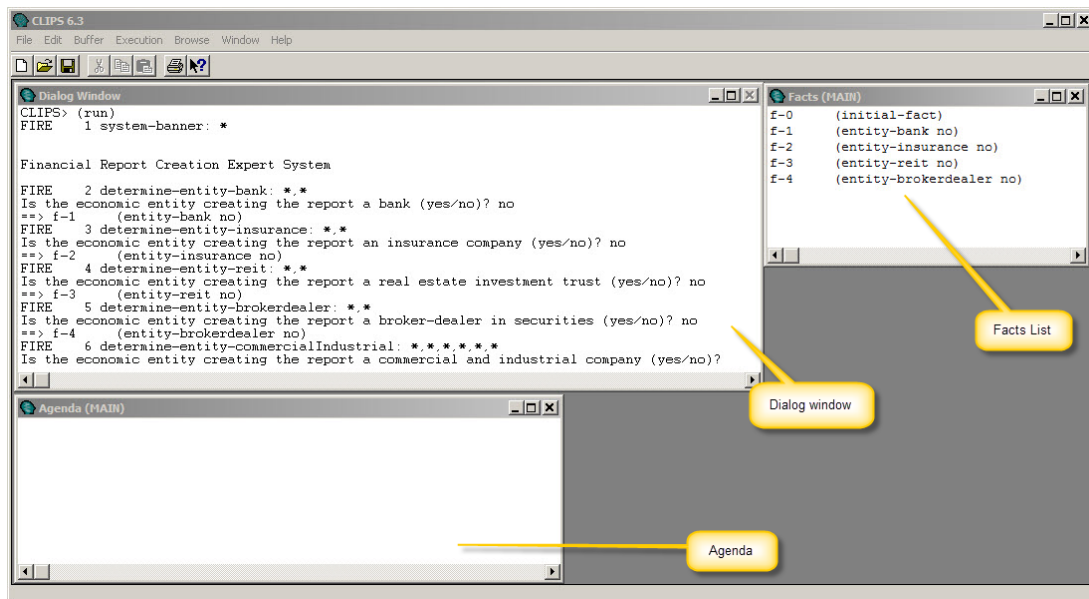
While CLIPS is not a tool the average professional accountant will use, CLIPS does offer a way to understand how expert systems work.

Recognize that CLIPS is a system for building any expert system that you want. As such, you have to understand how to put the pieces of an expert system together. Professional accountants will not have to do this for expert systems which create financial reports. Just as architects don't have to build computer aided design or computer aided manufacturing expert systems from scratch, neither will accountants. Software developers will create expert systems that professional accountants will use.

However, some professional accountants will build expert systems. Tools such as CLIPS allow someone who understand logic programming to create expert systems for other domains. For example, a small business might have a task they perform manually which they might want replaced by an automated expert system. Professional accountants will help small businesses create such micro expert systems. This could be a niche service offered by professional accountants.

#### 3.3.1. Overview of CLIPS

Consider the software application interface of CLIPS below:



The interface is showing three parts:

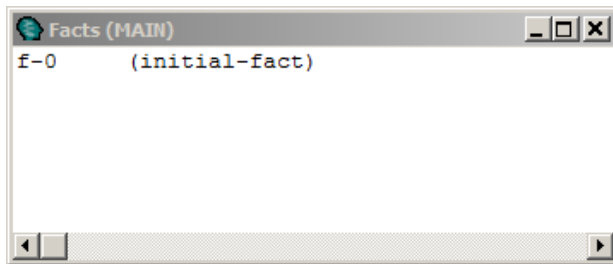
<sup>247</sup> Using CLIPS to Understand Expert Systems and Logic Programming, <http://xbrl.squarespace.com/journal/2016/9/15/using-clips-to-understand-expert-systems-and-logic-programmi.html>

- **Fact list:** In the upper right hand corner you see a facts list or “database of facts”.
- **Agenda:** In the lower left side you see an agenda window. Currently the agenda window is empty which means the expert system has no more tasks to complete.
- **Dialog window:** In the upper left, the largest window is the dialog window. The dialog window is where the user interacts with the expert system software.

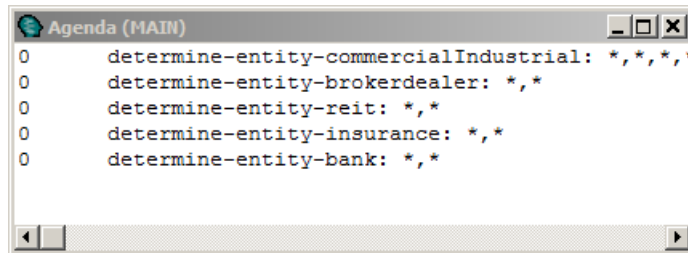
Remember that CLIPS is a universal programming language that is designed to enable anyone with the necessary skills to build any sort of expert system. That is why the interface is general. So please don't get distracted by the nature of the interface. Focus on the logic of how an expert system works.

When the expert system starts, the facts list and the agenda look as follows:

Facts list:



Agenda:



The fact list is empty (except for a default fact) and then the agenda has five activations. Basically, no facts are known (in the case of this system) and there are five items on the agenda to determine the accounting activity of the economic entity creating the financial report.

This is the terminology used by the CLIPS system for building expert systems which will give you an idea of how you interact with CLIPS. CLIPS uses a forward chaining problem solving method.

- **Goal:** The system will cease execution when no *activations* are on the *agenda*.
- **Strategy:** High-level plan for achieving a goal.

- **Fact:** A fact is the same as an XBRL definition of a fact.
- **Fact list:** A fact list is the set of facts the system is currently working with to arrive at some goal.
- **Rule:** A rule is a relation between facts or a relation between a fact and the characteristics or traits of a fact.
- **Assertion:** Assert a new fact.
- **Retraction:** Retract an existing fact.
- **Activation:** An activation is a rule that is active because it matches the forward chaining strategy.
- **Agenda:** The agenda is a collection of activations which are those rules which match pattern entities. Zero or more activations may be on the agenda.
- **Salience:** When multiple activations are on the agenda, the system automatically determines which activation is appropriate to fire. The system orders the activations on the agenda in terms of increasing priority or salience.
- **Depth strategy:** In the depth strategy, new activations are placed on the agenda after activations with higher salience, but before activations with equal or lower salience. All this simply means is that the agenda is ordered from highest to lowest salience.
- **Conflict resolution:** The inference engine sorts the activations according to their salience. This sorting process is called conflict resolution because it eliminates the conflict of deciding which rule should fire next.
- **Refraction:** Refraction is the management of when rules fire so trivial loops are avoided. Without refraction, expert systems always would be caught in trivial loops. That is, as soon as a rule fired, it would keep firing on that same fact over and over again. In the real world, the stimulus that caused the firing eventually would disappear.
- **Clear:** Removes all facts and rules from working memory, basically resetting the system.

Note that Jess<sup>248</sup> is a version of CLIPS written in Java.

### ***3.4. Opportunities for Using Expert Systems in Financial Reporting***

What type of expert systems could be useful in the process of creating a financial report? Why would you even want to create an agent for financial reporting<sup>249</sup>?

#### ***3.4.1. Starting simple; example of one type of expert system***

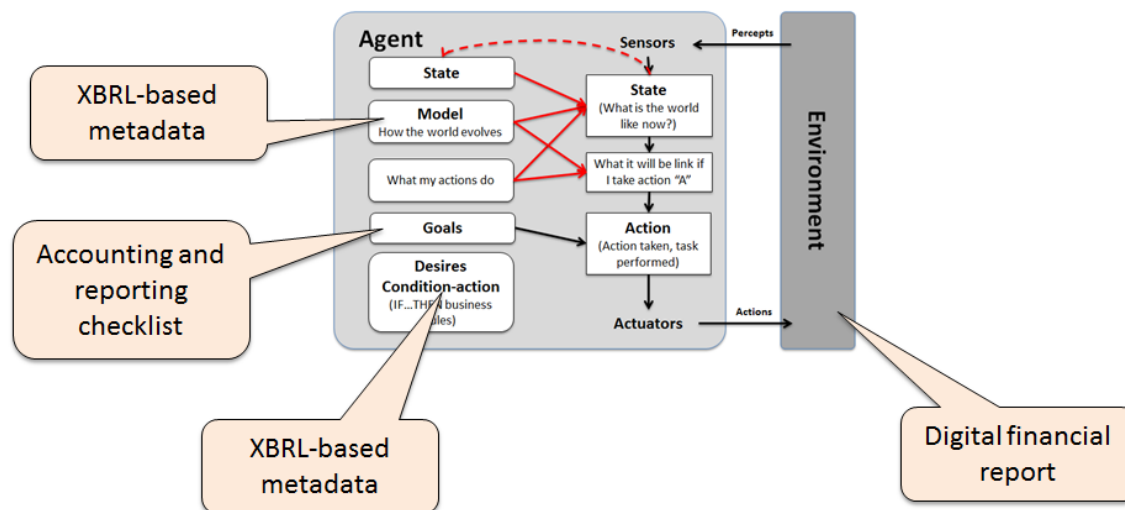
The following is the architecture of an agent that could be helpful in the process of creating a financial report<sup>250</sup>. I believe that such an expert system would be

---

<sup>248</sup> Jessrules, *The Java Expert System Shell*, <https://jessrules.com/jess/docs/45/>

<sup>249</sup> *Benefits Offered by an Expert System*, Retrieved July 24, 2016; <http://xbrl.squarespace.com/journal/2016/5/30/understanding-the-benefits-offered-by-expert-systems.html>

classified as a global standard rational, deliberative, non-learning, goal-based intelligent software agent.



### 3.4.2. Automating accounting and reporting checklists

Most accountants are familiar with the disclosure checklist. They use that human readable checklist as a memory jogger in the process of creating financial reports. What if you made that checklist also readable by machines and what if financial reports were structured? Automating the disclosure checklist will be one of the first uses of intelligent agent software<sup>251</sup>. This will not be a batch process that you run when a financial report is complete; rather it will be an expert system intelligent software agent watching over you as you create the financial report.

### 3.4.3. World's first expert system for creating financial reports

I believe that a software developer and I have created what we can honestly call the world's first expert system for creating financial reports (as far as I am aware)<sup>252</sup>.

But what is even more interesting is that what drives this expert system is a global standard XBRL-based general purpose business reporting expert system. The system is both in the form of a tool that is very approachable by business professionals and an API interface.

While the software application is admittedly rudimentary, it does successfully prove the concept of using an expert system in the process of creating a financial report.

Please watch this YouTube channel that will be updated to help explain this expert system for creating digital financial reports:

<https://www.youtube.com/channel/UCRIbipm3f0DaGPuLK51rvHA>

<sup>250</sup> Automating Accounting and Reporting Checklists, Retrieved July 24, 2016; <http://xbrl.squarespace.com/journal/2016/5/5/automating-accounting-and-reporting-checklists.html>

<sup>251</sup> Automating Accounting and Reporting Checklists, Retrieved July 24, 2016; <http://xbrl.squarespace.com/journal/2016/5/5/automating-accounting-and-reporting-checklists.html>

<sup>252</sup> World's First Expert System for Creating Financial Reports, <http://xbrl.squarespace.com/journal/2017/4/27/worlds-first-expert-system-for-creating-financial-reports.html>

### 3.5. *Putting the Expertise into an XBRL-based Knowledge Based System for Creating Financial Reports*

One type of practical knowledge is know-how; how to accomplish something.

Creating a knowledge based system involves the transformation of machine-readable instructions in such a way as to explain to a machine how a system works and how to make a system work the way you want that system to work.

Then, brick-by-brick, much like building a house, business domain experts and software engineers can create tools that automate certain types of tasks in that process. Humans encode information, represent knowledge, and share meaning using machine-readable patterns, languages, and logic. That will be the way an increasing number of work tasks will be performed in the Digital Age of accounting, reporting, and auditing. The result will be more efficient processes.

The document *Putting the Expertise into an XBRL-based Knowledge Based System for Creating Financial Reports*<sup>253</sup> explains how a software engineer and I created proof of concept to test the feasibility of such a system.

## 4. Introduction to Intelligent Software Agents

This section provides a comprehensive introduction to intelligent software agents for professional accountants.

This is a dangerous time for professional accountants. Technology is changing very rapidly<sup>254</sup>. For example, terms like “Big Data” and “Deep Learning” and “Artificial Intelligence” are being thrown about by software vendors, some of which are little more than snake oil salesman; with these new technologies comes a significant amount of hype. On the other hand, many of these software vendors have very useful products what will transform financial reporting as it is practiced today. Knowledge is very useful when one is trying to differentiate the hype from real possibilities.

Setting the right expectations is another big challenge. This example will help you understand what I mean. There is a lot of good and bad information relating to driverless cars. Tesla has functionality in its cars that is similar to driverless car-type functionality. Imagine trying to set a goal for what you want such a car to achieve:

- **Goal A:** Create a car that can travel anywhere in the United States without a driver.
- **Goal B:** Create a car that can assist the driver to drive anywhere in the United States.
- **Goal C:** Create a car that can assist a driver when driving on an Interstate Highway.

---

<sup>253</sup> *Putting the Expertise into an XBRL-based Knowledge Based System for Creating Financial Reports*, <http://pesseraact.azurewebsites.net/PuttingTheExpertiseIntoKnowledgeBasedSystem.pdf>

<sup>254</sup> *AICPA News Update: Technologies are poised to reshape the accounting landscape*, <http://xbri.squarespace.com/journal/2017/7/7/aicpa-news-update-technologies-are-poised-to-reshape-the-acc.html>

What is the relative cost of achieving goals A, B, and C above? What is the relative probability of even being able to achieve goals A, B, and C without significant, expensive research and development?

The point is that setting the right expectations helps one understand what is actually practical and useful. Thinking that technology will have no impact on how you perform your work will be a complete disaster for your career as an accountant and/or for your business.

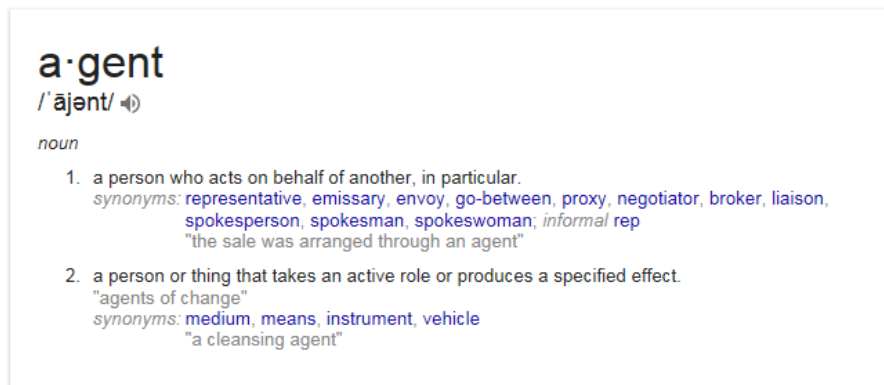
A core part of expert systems and other knowledge based systems is the intelligent software agent. This document helps professional accountants sort through all the information and misinformation that they are hearing related to XBRL-based structured digital financial reporting and helps them understand what an intelligent software agent is, basics on how they work, and examples of how they might be used.

## 4.1. Deconstructing the Notion of an Intelligent Software Agent

To understand intelligent software agents one first needs to understand the notion of an agent. This section is dedicated to setting your perspective as to the notion of an agent. The section provides specific definitions, deconstructing the pieces so that we can subsequently put the pieces back together.

### 4.1.1. Definition of an agent

Google defines agent<sup>255</sup> as “a person who acts on behalf of another” and “a person or thing that takes an active role or produces a specified effect”.



An agent performs specific tasks on behalf of another.

Important to the definition of agent is the law of agency<sup>256</sup>. Some principle, or the person making use of the agent, authorizes the agent to work on their behalf to perform an agreed upon specific task. The important key points are the *authority* to act and an *agreement* on the specific task or tasks that the agent is supposed to perform.

Business professionals generally understand the notion of an agent because they get introduced to this notion in their Business Law 101 class in college. This class

<sup>255</sup> Google search, *Agent definition*, retrieved August 14, 2016, <https://www.google.com/search?q=agent+definition>

<sup>256</sup> Wikipedia, *Law of agency*, retrieved August 14, 2016, [https://en.wikipedia.org/wiki/Law\\_of\\_agency](https://en.wikipedia.org/wiki/Law_of_agency)

introduces them to the notion of human agents. But what if the agent is not a human?

#### **4.1.2. Software agents**

In the case of computer science, an agent is a software program. Wikipedia defines software agent<sup>257</sup> (paraphrasing) as “a computer program that acts for a user or other program in a relationship of agency to perform some action”.

Software agents are sometimes also referred to as “bots” from the term robot.

#### **4.1.3. Intelligent agents**

An intelligent agent<sup>258</sup> is an abstract notion that links the *real world* agent and the notion of agency with an *implementation* of that functionality within software. An intelligent agent is the abstract functionality of a system similar to a computer program; it is not the computer software program itself.

Intelligent agent, as we are using it, is an idea related to artificial intelligence. An intelligent agent is an autonomous entity which observes its environment through sensors and acts upon that environment using actuators in the pursuit of some goal. Intelligent is a software design philosophy.

#### **4.1.4. Intelligent software agents**

The main difference between a software agent and an ordinary program is that a software agent is autonomous; that is, it must operate without direct intervention of humans or others.

Intelligent software agents are engineered using specific principles<sup>259</sup>.

#### **4.1.5. Notion of an intelligent software agent which we will use**

An agent is an entity capable of sensing the state of its environment and acting upon it based on a set of specified rules.

The consultancy firm McKinsey predicts that intelligent software agent use will increase in business<sup>260</sup>. Use of intelligent software is expected to be the norm in law firms<sup>261</sup> by the year 2020.

## **4.2. Understanding what Intelligent Software Agents Do**

This section helps you understand what intelligent software agents do and how they do it.

---

<sup>257</sup> Wikipedia, *Software agent*, retrieved August 14, 2016, [https://en.wikipedia.org/wiki/Software\\_agent](https://en.wikipedia.org/wiki/Software_agent)

<sup>258</sup> Wikipedia, *Intelligent agent*, retrieved August 14, 2016, [https://en.wikipedia.org/wiki/Intelligent\\_agent](https://en.wikipedia.org/wiki/Intelligent_agent)

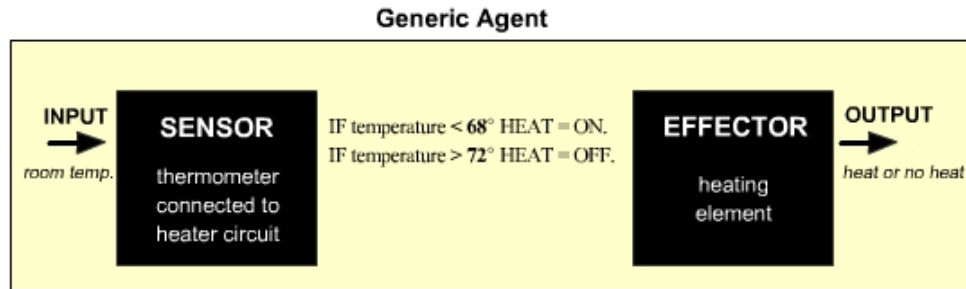
<sup>259</sup> Frances M.T. Brazier, Catholijn M. Jonker, Jan Treur Vrije, *Principles of Component-Based Design of Intelligent Agents*, <http://www.few.vu.nl/~wai/Papers/DKE02.princ.pdf>

<sup>260</sup> McKinsey, *Artificial Intelligence Meets the C-Suite*, <http://www.mckinsey.com/business-functions/strategy-and-corporate-finance/our-insights/artificial-intelligence-meets-the-c-suite>

<sup>261</sup> Managing Partner, *Artificial intelligence to be 'the norm' in law firms by 2020*, <http://www.managingpartner.com/news/business-strategy/artificial-intelligence-be-%E2%80%98-norm%E2%80%99-law-firms-2020>

#### 4.2.1. Basic example of an intelligent software agent

A basic example of an intelligent software agent is a basic thermostat<sup>262,263</sup>.



**Fig. 1:** Thermostat agent

Software reads the current room temperature using a sensor and uses a set of rules to turn the heat on or turn the heat off to impact the future room temperature in the room or environment in which the sensor is located.

#### 4.2.2. Basic model of an intelligent software agent

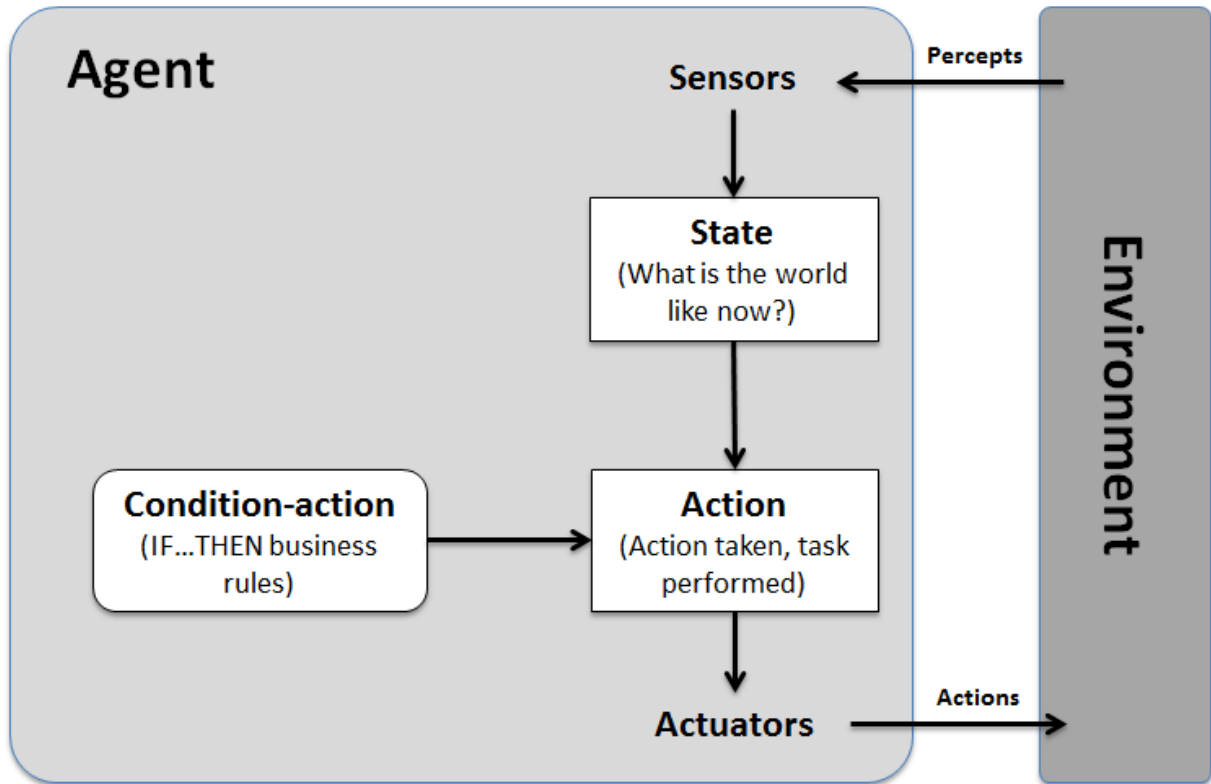
If you look at the basic intelligent software agent above and then you look at lots of other tasks that such intelligent software agents do you see patterns. Intelligent software agents leverage these patterns. Intelligent software agents are computer code written in a specific way. An **agent** is an entity capable of **sensing** the **state** of its **environment** and **acting** upon it based on a **set of specified rules**. Remember that an agent performs specific tasks on behalf of another in order to achieve some agreed upon goal established by the principal that employs the agent and the agent which will act in behalf of the principle.

In the case of software, an agent is a software program. Consider that definition of an agent and look at the graphic below to get an idea of how intelligent agent software works:

<sup>262</sup> Consortium of Cognitive Science Instruction, *Introduction to Intelligent Agents*, Retrieved July 24, 2016; [http://www.mind.ilstu.edu/curriculum/ants\\_nasa/intelligent\\_agents.php](http://www.mind.ilstu.edu/curriculum/ants_nasa/intelligent_agents.php)

<sup>263</sup> Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach (Third edition)*, page 31, <http://aima.cs.berkeley.edu/>





An intelligent agent is software that assists people and acts on their behalf. Intelligent agents work by allowing people to:

- delegate work that they could have done to the agent software,
- perform repetitive tasks,
- remember things you forgot,
- intelligently find, filter and summarize complex information,
- customize information to your preferences,
- learn from you and even make recommendations to you.

#### **4.2.3. Business rules drive intelligent software agents and expert systems automating work**

An example of one major shift is provided by what professional accountants call the "disclosure checklist". Accountants creating financial reports often use accounting and reporting checklists or "disclosure checklists" as memory joggers to help them get the reports right<sup>264</sup>. These memory joggers were created to be read by humans and can be a couple hundred pages. What if a financial report was structured, such as an XBRL-based public company financial report that must be submitted to the U.S. Securities and Exchange Commission. What if these human-readable memory joggers could be made machine-readable. And what if an intelligent software agent

<sup>264</sup> *Automating Accounting and Reporting Checklists*, <http://xbrl.squarespace.com/journal/2016/5/5/automating-accounting-and-reporting-checklists.html>

could be created to automate the manual task of checking a financial report to make sure that report was mechanically correct.

Note the statement “mechanically correct”. This is a very important distinction. No computer program will ever have the judgement of a professional accountant. See the section *Setting the right expectations* later in this document. But computer programs can perform work if the financial report is structured and the necessary business rules are made machine-readable. How much of a disclosure checklist can be automated? That percentage is as-of-yet to be determined. Perhaps 20% can be automated or even 80% will be automated. Maybe even a higher percentage. The probability 0% of a disclosure checklist can be automated is extremely low.

Humans augmented by machine capabilities, much like an electronic calculator enabling a human to do math quicker, will empower knowledge workers who know how to leverage the use of those machines.

#### **4.2.4. Important terminology related to intelligent software agents**

The following is a summary of important terminology<sup>265</sup> that is used to discuss intelligent software agents.

- **Agent:** An **agent** is an entity capable of **sensing** the **state** of its **environment** and **acting** upon it based on a **set of specified rules**. Remember that an agent performs specific tasks on behalf of another in order to achieve some agreed upon goal established by the principal that employs the agent and the agent which will act in behalf of the principle.
- **Environment:** environment in which the agent operates; description of the state of affairs that change over time as real world situations do.
- **Sensing capabilities:** capability of the agent to understand its environment; determines the sort of data the agent is capable of receiving as input.
- **Percept:** A percept refers to the agent's perceptual inputs at any given moment.
- **Percept sequence:** The percept sequence represents the complete sequence of percepts the agent has sensed or perceived during his lifetime.
- **State:** current conditions of the environment. Or, a projected conditions of the environment based on actions taken.
- **Actions:** change in the environment brought about by the agent, requiring the agent to update its model of the world, which in turn may cause the agent to change its immediate intention.
- **Condition-action rules:** A formal and implementable expression of some business user requirement. Includes definitions of terms, structural assertions, action assertions, and derivations.
- **Desires:** overall policies or goals of the agent.
- **Action selection architecture:** an agent decides what to do next by consulting both its internal state, the state of the world, and its current goal; then the agent uses decision making procedures to select an action.

---

<sup>265</sup> Consortium of Cognitive Science Instruction, *Introduction to Intelligent Agents, Taxonomy of agents*, Retrieved July 24, 2016; [http://www.mind.ilstu.edu/curriculum/ants\\_nasa/intelligent\\_agents.php](http://www.mind.ilstu.edu/curriculum/ants_nasa/intelligent_agents.php)

#### 4.2.5. Agents Working Together: Multi-agent Systems

Note that one agent must be able to communicate with other agents and exchange information between agents. Below is terminology related to the communication between agents<sup>266</sup>.

- **Multi-agent system:** When an agent coexists in an environment with other agents, perhaps collaborating or competing with them, the system is considered a multi-agent system.
- **Coalition:** A coalition is any subset of agents in the environment.
- **Strategy:** A strategy is a function that receives the current state of the environment and outputs the action to be executed by a coalition.
- **Blackboard:** A blackboard structure is a communication form that consists of a shared resource divided into different areas of the environment where agents can read or write any significant information for their actions.
- **Coordination:** Coordination is essential in multi-agent system because it provides coherency to the system behavior and contributes to achieving team or coalition goals.
- **Cooperation:** Cooperation is necessary as a result of complementary skills and the interdependency present among agent actions and the inevitability of satisfying some success criteria.
- **Competition:** Another possible model is that in which the agents are self-motivated or self-interested agents because each agent has its own goals and might enter into competition with the other agents in the system to achieve these goals. In this sense, competition might refer to accomplishing or distributing certain tasks.
- **Negotiation:** Negotiation might be seen as the process of identifying interactions based on communication and reasoning regarding the state and intentions of other agents.

When one intelligent software agent communicates with another intelligent software agent some technical syntax and semantics must be employed<sup>267</sup>. The XBRL global standard can be used for such communication and collaboration.

---

<sup>266</sup> C# - Applying AI to a Multi-Agent 'Mini-Basketball' Game, Retrieved July 24, 2016; <https://msdn.microsoft.com/en-us/magazine/mt736456.aspx>

<sup>267</sup> Hanh Tran & Thoavy Tran, *Intelligent Agent, Agent Communication Languages*, [http://groups.engin.umd.umich.edu/CIS/course.des/cis479/projects/agent/Intelligent\\_agent.html#dep6](http://groups.engin.umd.umich.edu/CIS/course.des/cis479/projects/agent/Intelligent_agent.html#dep6)

### 4.3. *Building your Understanding of Intelligent Software Agents*

We will build your understanding of intelligent software agents piece-by-piece in this section.

#### 4.3.1. *Artificial intelligence*

Artificial intelligence is a branch of computer science. There are many good descriptions of artificial intelligence<sup>268</sup>. Here is one good definition:

Artificial intelligence is the automation of activities that we associate with human thinking and activities such as decision making, problem solving, learning and so on.

Those trying to make artificial intelligence work over the past 40 or so years have had limited success. But that is changing. Both under estimating or over estimating the capabilities the computer software will be able to achieve can have catastrophic consequences.

There are two types of artificial intelligence, specialized and generalized:

- **Specialized:** An example of specialized artificial intelligence is programming a computer to play chess. The software performs one specific task. Specialized artificial intelligence is fairly easy to achieve.
- **Generalized:** An example of generalized artificial intelligence is the sort of science-fiction stuff you see in the movies. Why do you only see this in the movies? Because generalized artificial intelligence is extremely hard to make work.

#### 4.3.2. *Expert systems*

Expert systems<sup>269</sup> is a branch of artificial intelligence. The following is a definition of an expert system:

Expert systems are computer programs that are built to mimic human behavior and knowledge. The computer program performs tasks that would otherwise be performed by a human expert. A model of the expertise of a domain of knowledge of the best practitioners or experts is put into machine-readable form and the expert system reaches conclusions or takes actions based on that information.

#### 4.3.3. *Business rules*

Key to employing artificial intelligence and therefore making an expert system or intelligent software agent<sup>270</sup> work is business rules<sup>271</sup> of the domain being put into machine-readable form.

---

<sup>268</sup> AlanTuring.net, *What is Artificial Intelligence?*, [http://www.alanturing.net/turing\\_archive/pages/reference%20articles/What%20is%20AI.html](http://www.alanturing.net/turing_archive/pages/reference%20articles/What%20is%20AI.html)

<sup>269</sup> *Understanding the Components of an Expert System*, <http://xbrl.squarespace.com/journal/2016/5/24/understanding-the-components-of-an-expert-system.html>

<sup>270</sup> Wikipedia, *Intelligent Agent*, retrieved July 24, 2016; [https://en.wikipedia.org/wiki/Intelligent\\_agent](https://en.wikipedia.org/wiki/Intelligent_agent)

<sup>271</sup> *Comprehensive Introduction to Business Rules*, [http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.4\\_ComprehensiveIntroductionToBusinessRules.pdf](http://xbrl.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.4_ComprehensiveIntroductionToBusinessRules.pdf)

#### ***4.3.4.Important distinctions between intelligent software agents***

There are two important distinctions that you should keep in the back of your mind when we talk about intelligent software agents:

- **Rational agent:** A rational agent is one that acts so as to achieve the best outcome or, when there's uncertainty, the best expected outcome. Rationality as used here refers to following the rules of logical reasoning, making correct inferences, and selecting the appropriate action that will lead to achieving the desired goal.
- **Autonomous agent:** An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.

Every intelligent software agent that we are interested in is generally both a rational and autonomous agent.

#### ***4.3.5.Categories of intelligent software agent functionality***

The functionality of an intelligent software agent can be classified into groups, or categories:

- **Reactive agent:** A reactive agent is capable of maintaining an ongoing interaction with the environment and responding in a timely fashion to changes that occur in it.
- **Pro-active agent:** A pro-active agent is capable of taking the initiative; not driven solely by events, but capable of generating goals and acting rationally to achieve them.
- **Deliberative agent:** A deliberative agent symbolically represents knowledge and makes use of mental notions such as beliefs, intentions, desires, choices and so on. This is implemented using a belief-desire-intension model.
- **Hybrid agent:** A hybrid agent is one that mixes some of all the different architectures.

Every intelligent software agent falls into one of those four categories.

#### ***4.3.6.Categories of intelligent software agent sophistication***

Intelligent software agents can be grouped as to the level of sophistication offered by the agent:

- **Generic agent:** An agent is anything that perceives an environment through sensors and acts or reacts upon the environment through effectors.
- **Simple reflex agent:** A simple reflex agent looks up what it should do from a list of rules in response to its perception to the environment.
- **Model-based reflex agent:** A model-based reflex agent is the same thing as a simple reflex agent except that a model-based reflex agent has a model of how the environment evolves.
- **Goal-based agent:** A goal-based agent has a goal or set of goals that it actively pursues in accordance with an agenda (so this type of agent is proactive, not just reactive). A goal based agent has a representation of the

current state of the environment and how that environment works. The agent pursues policies or goals that may not be immediately attainable. And so, goal based agents do not live merely in the moment. These agents consider different scenarios before acting on their environments, to see which action will probably attain a goal. This consideration of different scenarios is called search and planning.

- **Utility-based agent:** A utility-based agent is a more sophisticated type of goal-based agent that also rates each possible scenario to see how well it achieves certain criteria with regard to production of the good outcome, therefore it is more adaptive. A utility measure is applied to the different possible actions that can be performed in the environment. The utility-based agent will rate each scenario to see how well it achieves certain criteria with regard to the production of a good outcome. Things like the probability of success, the resources needed to execute the scenario, the importance of the goal to be achieved, the time it will take, might all be factored in to the utility function calculations.

Every intelligent software agent falls into one of those groups in terms of sophistication of functionality offered.

#### **4.3.7. Learning versus non-learning intelligent software agents**

Every intelligent software agent can be either a learning or non-learning agent:

- **Learning agent:** A learning agent is one that requires some training to perform well, adapts its current behavior based on previous experiences and evolves over time.
- **Non-learning agent:** A non-learning agent is one that doesn't evolve or relate to past experiences and is hard coded and independent of its programming.

A learning agent is significantly more sophisticated in terms of functionally, but the tradeoff is that they are also significantly more complex to create.

### **4.4. Intelligent software agents assisting humans**

Artificial intelligence is the automation of activities that we associate with human thinking and activities such as decision making, problem solving, learning and so on<sup>272</sup>.

Expert systems is a branch of artificial intelligence. An intelligent agent is software that assists people and acts on their behalf. Intelligent agents work by allowing people to:

- delegate work that they could have done to the agent software,
- perform repetitive tasks,
- remember things you forgot,
- intelligently find, filter and summarize complex information,
- customize information to your preferences,

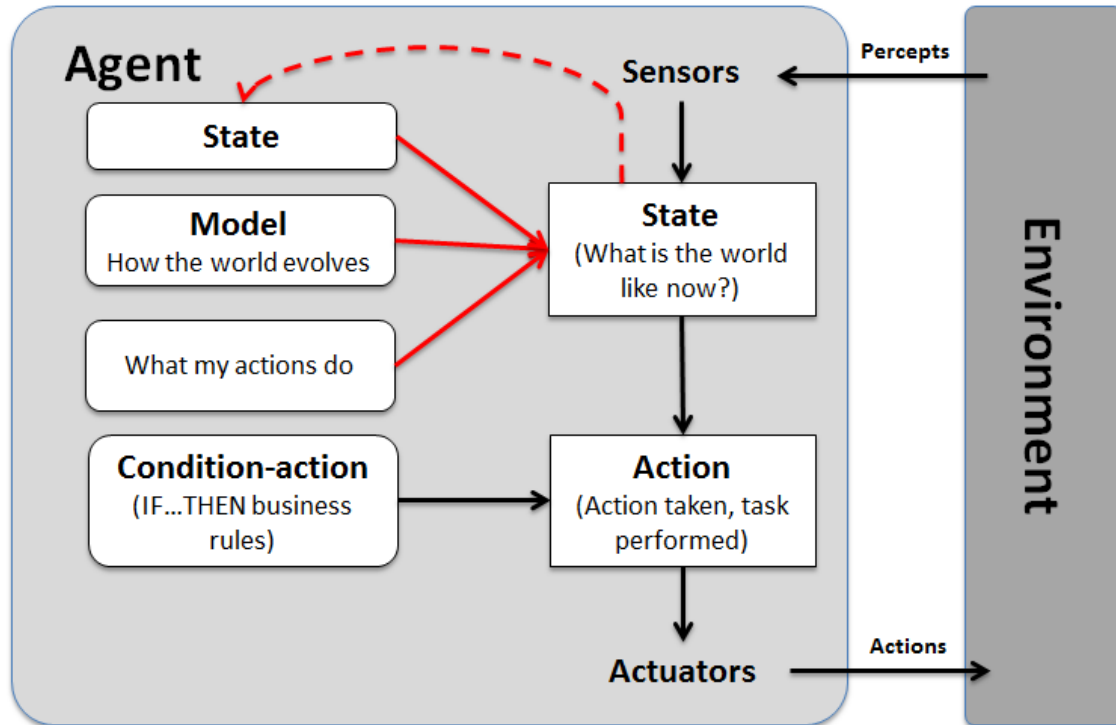
---

<sup>272</sup> *Introduction to Artificial Intelligence Terminology*, <http://xbrl.squarespace.com/journal/2016/7/21/introduction-to-artificial-intelligence-terminology.html>

- learn from you and even make recommendations to you.

An **agent** is an entity capable of **sensing** the **state** of its **environment** and **acting** upon it based on a set of specified **rules**. An agent performs specific tasks on behalf of another. In the case of software, an agent is a software program.

The main difference between a software agent and an ordinary program is that a software agent is autonomous; that is, it must operate without direct intervention of humans or others. There are many different types of intelligent software agents<sup>273</sup>.



Intelligent agents can perform sophisticated work. A rational agent is one that acts so as to achieve the best outcome or, when there's uncertainty, the best expected outcome. Rationality as used here refers to following the rules of logical reasoning, making correct inferences, and selecting the appropriate action that will lead to achieving the desired goal.

Machine-readable business rules are key to creating intelligent software agents that provide the functionality within an expert system.

#### 4.4.1. Getting the thick layer of metadata

No one disputes that you need a thick layer of metadata to get computers to perform useful work for you<sup>274</sup>. The more metadata you have, the more a computer can do. What can be in dispute, particularly if you are not aware of the differences between

<sup>273</sup> Introduction to Intelligent Agents for Business Professionals (DRAFT), [http://xbrl.site.azurewebsites.net/2016/Library/02\\_IntroducingIntelligentAgents.pdf](http://xbrl.site.azurewebsites.net/2016/Library/02_IntroducingIntelligentAgents.pdf)

<sup>274</sup> Understanding the Need for a Framework and Theory, <http://xbrl.squarespace.com/journal/2015/9/20/understanding-the-need-for-a-framework-and-theory.html>

the approaches, is the different ways you can get that thick layer of metadata. Here are the two ways:

- **Have the computer figure out what the metadata is:** This approach uses artificial intelligence, machine learning, deep learning, neural networks, and other high-tech approaches<sup>275</sup> to detecting patterns and figuring out the metadata.
- **Tell the computer what the metadata is:** This approach leverages business domain experts and knowledge engineers to piece together the metadata so that the metadata becomes available.

Both approaches can be appropriate in the right situation. If the tolerance for error is low, like in a financial report, machine learning and neural networks generally don't work. What works really well is when you tell the computer what the metadata is first which serves as training data and then machines can be employed to leverage that training data to derive more metadata which human domain experts sort through and make right.

So again, both approaches can be useful in the right situation. Understanding the pros and cons of each approach helps you make good choices of which approach to employ. Technology has its limits which computer science professionals sometimes forget to mention to business professionals.

#### **4.4.2. Benefits offered by expert systems**

In the future, the accounting and reporting rules will exist in both human-readable and machine-readable form and will drive the expert systems and intelligent software agents which professional accountants use to create financial reports. Benefits from the use of expert systems and intelligent software agents include:

- **Automation:** elimination of routine, boring, repetitive, mundane, mechanical tasks that can be automated
- **Consistency:** computers are good at performing repetitive, mechanical tasks without variation whereas humans are not; computers do not make mistakes and are good at repeating exactly the same thing each time
- **Diligence and tenacity:** computers excel at paying attention to detail; they never get bored or overwhelmed and they are always available and will keep doing their job until the task is complete with the same attention to detail
- **Reduced down-time:** computer based expert systems are tireless and do not get distracted
- **Availability:** such computer based expert systems are always available simultaneously in multiple places at one time; you get quick response times and can replace absent or scarce experts
- **Training:** the best practices of the best practitioners can be available to those that are new to and learning about a domain of knowledge
- **Longevity and persistence:** computer based expert systems do not change jobs or retire so knowledge gathered by an organization can remain within that organization

---

<sup>275</sup> *Understanding 'Deep Learning' and 'Neural Networks'*, <http://xbrl.squarespace.com/journal/2015/9/17/understanding-deep-learning-and-neural-networks.html>



- **Productivity:** computer based expert systems are cheaper than hiring experts and costs can be reduced at the same time that quality increases resulting in increased productivity
- **Multiple opinions:** Systems can integrate the view of multiple experts within a single system and choose between the preferred view of multiple expert opinions in the same system
- **Objectivity:** computers apply the same inductive and deductive logic consistently; emotion and personal preferences can be eliminated where they should be eliminated

Critical to understanding the sorts of tasks that expert systems will be capable of performing and should not or will never be able to perform takes the understanding of a domain professional. While computer based expert systems can effectively automate some work, this does not imply that these systems will automate all work or replace humans. They simply won't because they cannot. Computers are dumb beasts. There is a difference between subjectivity and objectivity; there is a difference between a mechanical task and a task requiring professional judgement. Professional accountants need to understand the difference<sup>276</sup>.

#### **4.4.3. Machines helping humans**

Machines are good at some things. Humans are good at other things. Understanding the difference is important for a lot of reasons. Machines can be made to perform certain tasks effectively and efficiently.

McKinsey published an article which predicted<sup>277</sup>:

“Many of the jobs that had once seemed the sole province of humans—including those of pathologists, petroleum geologists, and law clerks—are now being performed by computers.”

Understanding the true capabilities of computer software and humans will help you understand what skills you will want to have in order to make sure you can add value to your organization. Those same skills will help you use intelligent software agents appropriately.

Humans augmented by machine capabilities, much like an electronic calculator enabling a human to do math quicker, will empower knowledge workers who know how to leverage the use of those machines.

### **4.5. Details of Categories of Intelligent Software Agent Sophistication**

In the section *Categories of intelligent software agent sophistication* we summarized the different categories of intelligent software agents and the level of sophistication offered. In this section we want to provide more details on these different categories.

---

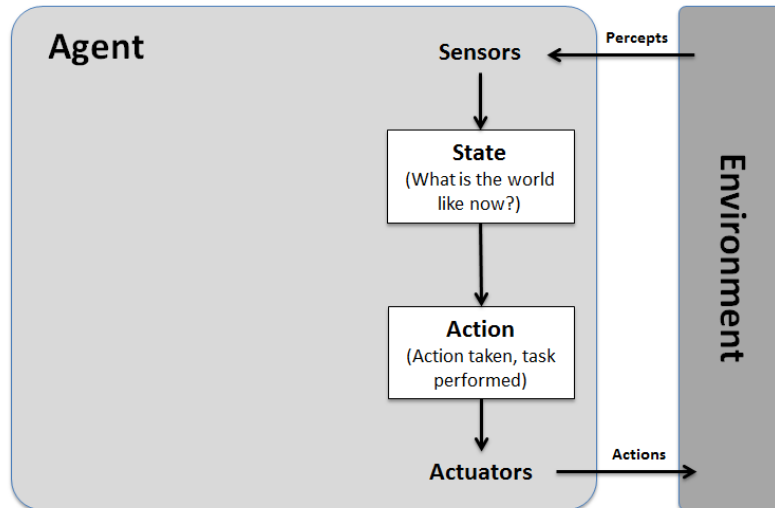
<sup>276</sup> Introduction to Knowledge Engineering for Professional Accountants, [http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01\\_Chapter02.3\\_KnowledgeEngineeringBasicsForProfessionalAccountants.pdf](http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.3_KnowledgeEngineeringBasicsForProfessionalAccountants.pdf)

<sup>277</sup> KcKinsey Quarterly 2014, *Artificial intelligence meets the C-suite*, <http://www.mckinsey.com/business-functions/strategy-and-corporate-finance/our-insights/artificial-intelligence-meets-the-c-suite>

#### 4.5.1. Generic agents

A generic agent<sup>278</sup> is anything that perceives an environment through sensors and acts or reacts upon the environment through effectors.

## Generic Agent



The following is the basic control loop of a generic agent:

While true

1. observe the world (**environment**);
2. update internal world model (**state**);
3. deliberate about what intention to achieve (**goal**);
4. use means/ends reasoning to get a plan for the intention (**condition-action rules**);
5. execute the plan (**action**)

End while

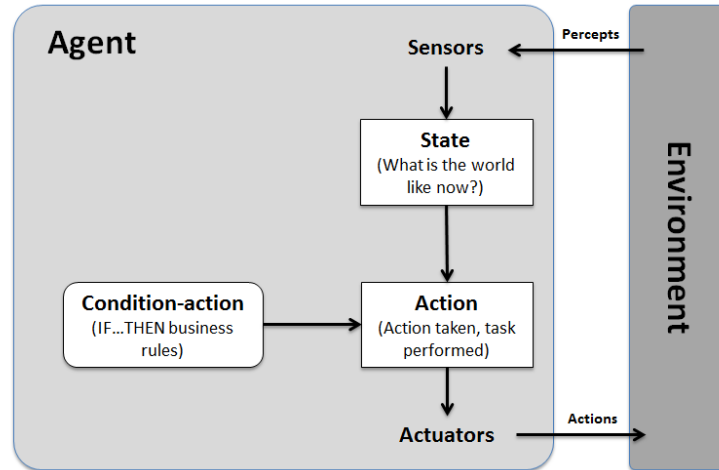
#### 4.5.2. Simple reflex agents

A simple reflex agent<sup>279</sup> looks up what it should do from a list of rules in response to its perception to the environment.

<sup>278</sup> Consortium of Cognitive Science Instruction, *Introduction to Intelligent Agents, Generic agent*, Retrieved July 24, 2016; [http://www.mind.ilstu.edu/curriculum/ants\\_nasa/intelligent\\_agents.php](http://www.mind.ilstu.edu/curriculum/ants_nasa/intelligent_agents.php)

<sup>279</sup> Wikipedia, *Intelligent Agents, Simple Reflex Agents*, retrieved August 14, 2016, [https://en.wikipedia.org/wiki/Intelligent\\_agent#Simple\\_reflex\\_agents](https://en.wikipedia.org/wiki/Intelligent_agent#Simple_reflex_agents)

## Simple Reflex Agent



A simple reflex agent looks up what it should do from a **list of rules** in response to its perception to the environment. This is the algorithm of simple reflex agent<sup>280</sup>:

Function Simple-Reflex-Agent (percept) returns action

- **persistent:** *rules*, a set of condition-action rules
- *state* <<<< Interpret-Input (*percept*)
- *rule* <<<< (Rule-Match (*state*, *rules*))
- *action* <<<< *rule*.Action
- **return** *action*

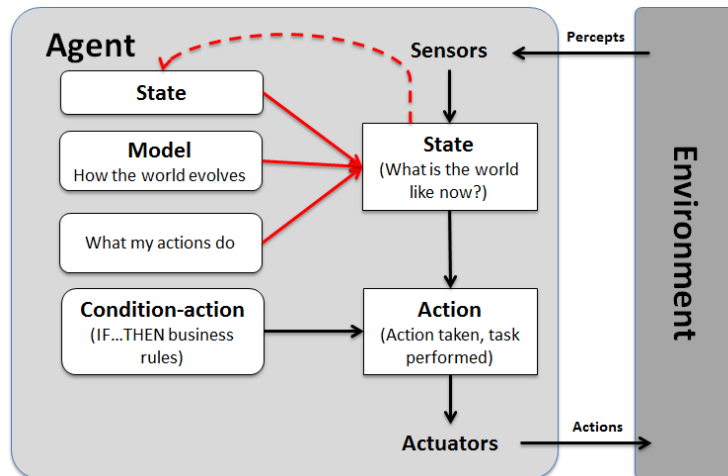
### 4.5.3. Model-based reflex agents

A model-based reflex agent<sup>281</sup> is the same thing as a simple reflex agent except that a model-based reflex agent has a model of how the environment evolves.

<sup>280</sup> Hugo Larochelle, *Intelligence Artificielle*, Retrieved July 24, 2016; <https://www.youtube.com/watch?v=TUHAVbaBLIq>

<sup>281</sup> Wikipedia, *Intelligent Agents, Model-based Reflex Agents*, retrieved August 14, 2016, [https://en.wikipedia.org/wiki/Intelligent\\_agent#Model-based\\_reflex\\_agents](https://en.wikipedia.org/wiki/Intelligent_agent#Model-based_reflex_agents)

## Model-based Reflex Agent

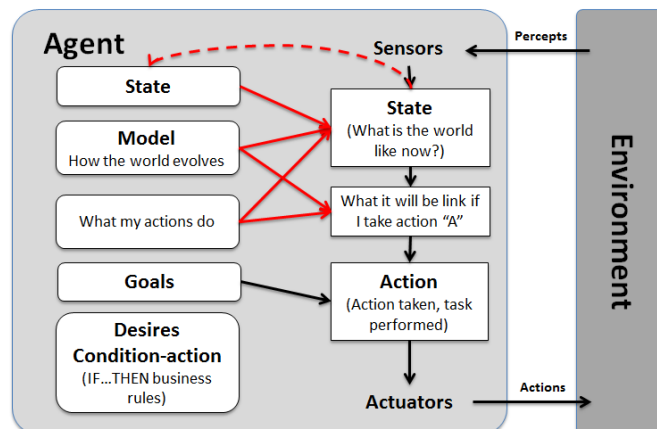


A model-based reflex agent is essentially the same as a simple reflex agent except that a model-based reflex agent adds the additional functionality of a model of the environment in which the agent operates.

### 4.5.4. Goal-based agents

A goal-based agent<sup>282</sup> has a goal or set of goals that it actively pursues in accordance with an agenda (so this type of agent is proactive, not just reactive). A goal based agent has a representation of the current state of the environment and how that environment works. The agent pursues policies or goals that may not be immediately attainable. And so, goal based agents do not live merely in the moment. These agents consider different scenarios before acting on their environments, to see which action will probably attain a goal. This consideration of different scenarios is called search and planning.

## Goal-based Agent



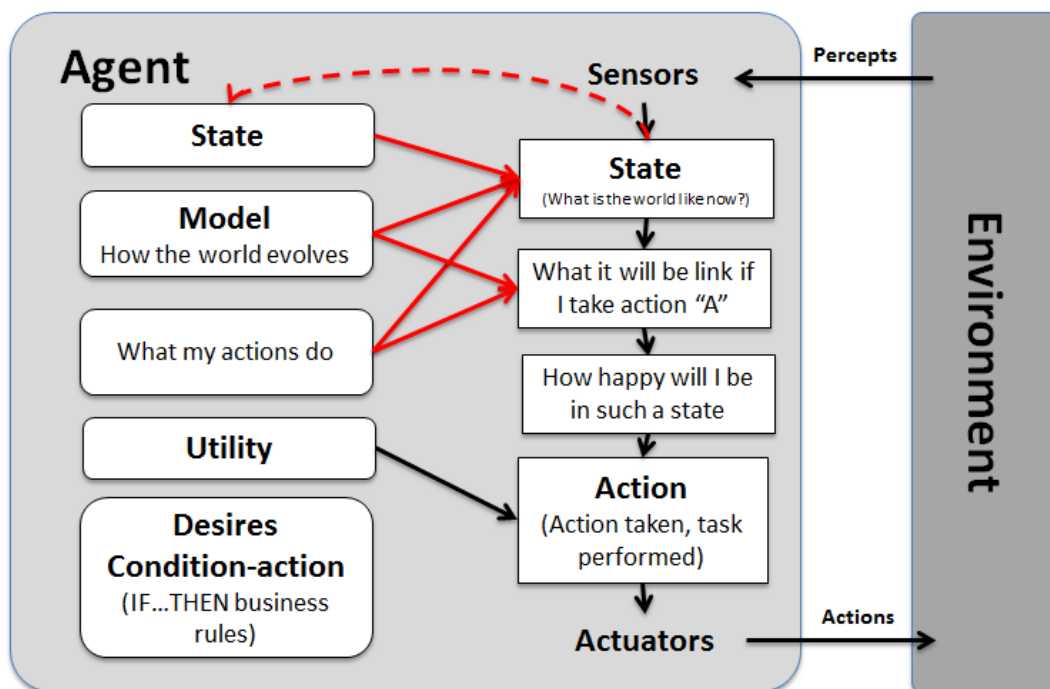
<sup>282</sup> Wikipedia, *Intelligent Agents, Goal-based Agents*, retrieved August 14, 2016, [https://en.wikipedia.org/wiki/Intelligent\\_agent#Goal-based\\_agents](https://en.wikipedia.org/wiki/Intelligent_agent#Goal-based_agents)

A goal-based reflex agent is essentially the same as a model-based reflex agent except that the goal-based reflex agent adds the additional functionality of wondering what the environment will be like if a specific action is taken, evaluating if that state is desirable or undesirable given specific goals, and the agent's desires given that state.

#### 4.5.5. Utility-based agents

A utility-based agent<sup>283</sup> is a more sophisticated type of goal-based agent that also rates each possible scenario to see how well it achieves certain criteria with regard to production of the good outcome, therefore it is more adaptive. A utility measure is applied to the different possible actions that can be performed in the environment. The utility-based agent will rate each scenario to see how well it achieves certain criteria with regard to the production of a good outcome. Things like the probability of success, the resources needed to execute the scenario, the importance of the goal to be achieved, the time it will take, might all be factored in to the utility function calculations.

## Utility-based Agent



A utility-based agent is the same as a goal-based reflex agent but adds additional functionality of wondering what the environment will be like if a specific action is taken, evaluating if that state is desirable or undesirable given specific goals and desires given that state, and evaluating how happy the agent will be within that state. Such agents might even be allowed to change its goals.

<sup>283</sup> Wikipedia, *Intelligent Agents, Utility-based Agents*, retrieved August 14, 2016, [https://en.wikipedia.org/wiki/Intelligent\\_agent#Utility-based\\_agents](https://en.wikipedia.org/wiki/Intelligent_agent#Utility-based_agents)

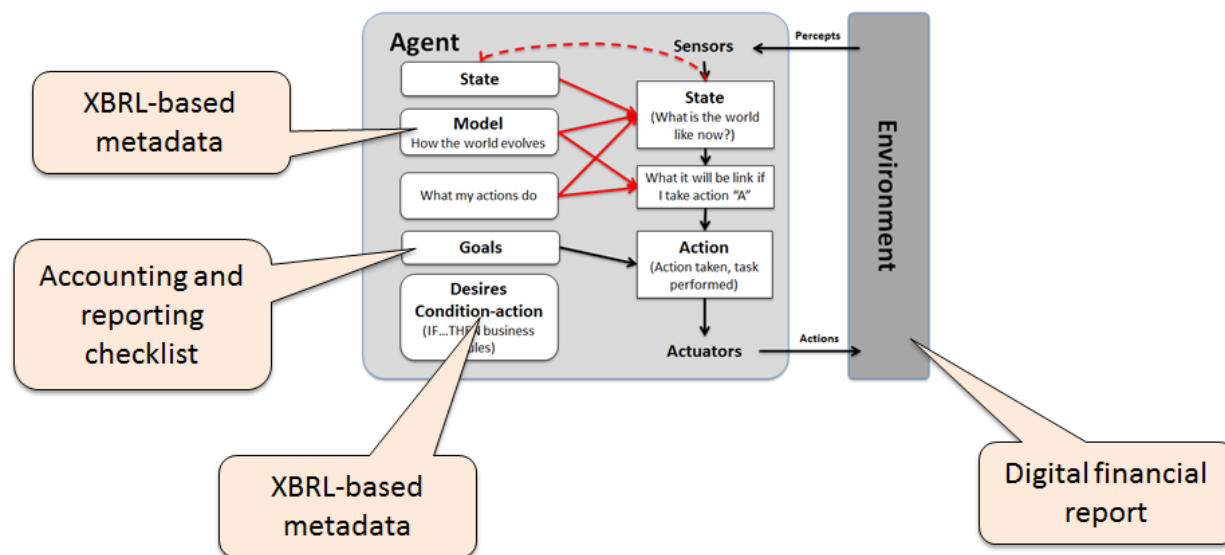
## 4.6. Using Intelligent Software Agents in Financial Reporting

So what is the best type of agent? That depends. What type of agents could be useful in the process of creating a financial report?

Why would you even want to create an agent for financial reporting<sup>284</sup>?

### 4.6.1. Starting simple; example of one type of agent

The following is the architecture of an agent that could be helpful in the process of creating a financial report<sup>285</sup>. I believe that such an agent would be classified as a global standard rational, deliberative, non-learning, goal-based agent.



### 4.6.2. Process robotics is disrupting accounting and finance

In the video, *Finance in a Digital World and the Impact on CFOs*<sup>286</sup>, John Steel who leads the finance transformation practice of Deloitte makes the statement,

"Five years from now there is either no CFO or the CFO is playing a different role."

John's view is consistent with what the AICPA and Journal of Accountancy are saying which is that technology is poised to change the accounting profession<sup>287</sup>.

In the video, John goes on to say, "Digital is having a tremendous impact and it's quite disruptive." In the video he goes over trends that are occurring. One of those trends is process robotics. John uses the term "lights out finance" meaning a finance

<sup>284</sup> *Benefits Offered by an Expert System*, Retrieved July 24, 2016; <http://xbrl.squarespace.com/journal/2016/5/30/understanding-the-benefits-offered-by-expert-systems.html>

<sup>285</sup> *Automating Accounting and Reporting Checklists*, Retrieved July 24, 2016; <http://xbrl.squarespace.com/journal/2016/5/5/automating-accounting-and-reporting-checklists.html>

<sup>286</sup> SAP, *Finance in a Digital World and the Impact on CFOs*, <http://events.sap.com/sapandasug/en/session/32172>

<sup>287</sup> *AICPA News Update: Technologies are poised to reshape the accounting landscape*, <http://xbrl.squarespace.com/journal/2017/7/7/aicpa-news-update-technologies-are-poised-to-reshape-the-acc.html>

that is completely automated. Now, we may never get to where the lights are completely out, but digital will involve automation of many existing manual processes.

A key word here is "disruptive". As pointed out by *The Innovators Dilemma*<sup>288</sup>, there are two types of innovation: sustaining and disruptive. *Sustaining innovation* meets customer's current needs, making incremental improvements in quality and efficiency of current processes. *Disruptive innovation* is about meeting future needs of customers.

Process robotics is about automating accounting, clerical, administrative, and other such tasks using software robots. Artificial intelligence technology drives these software robots. It really is a lot like how physical robots were employed to replace humans in manufacturing processes such as the process of building cars. Software robots cannot automate all tasks but certainly there are tasks which can be automated.

We don't want to overstate what innovations such as process robotics will provide. But we likewise don't want to understate the impact either. This is not about computers taking over the world. The way it will work is that you will have humans augmented by machine capabilities, much like an electronic calculator enabling a human to do math quicker, will empower professional accountants and others who know how to leverage those machines.

Underlying the process robotics will be intelligent software agents that interact with each other and with humans to get work done.

---

<sup>288</sup> YouTube, *The Innovator's Dilemma*, <https://www.youtube.com/watch?v=yUAtIQDllo8>