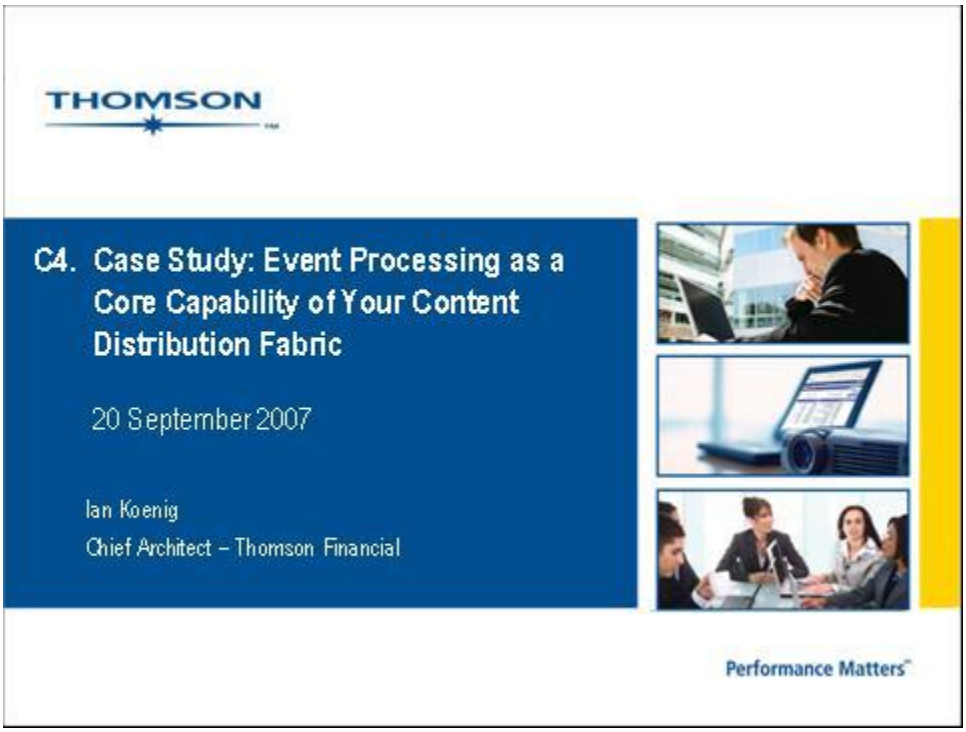
On 18 September 2007, I gave a presentation at the 3rd EPTS (Event Processing Technology Society) meeting and on 20 September I gave the same presentation at the Gartner Event Processing Symposium.

Following is the content of that presentation:



About the Author

Ian Koenig joined Thomson Financial in September 2005. Ian assumed the role of Chief Architect, reporting to the CTO, Bill Krivoshik. Ian has global responsibility for: Technology Strategy and Enterprise Architecture (information architecture, service oriented architecture (SOA), physical architecture, business architecture and solutions architecture) as well as architecture governance. Prior to joining Thomson, Ian was the Chief Architect for Products and Platforms at Reuters, where he worked for 19 years (beginning in 1986).

Throughout his 20+ year career, Ian Koenig has held technical positions of increasing scope and responsibility, beginning his career at Reveal Software on Long Island in 1982. Ian was one of seven people who received the Windows Pioneer Award in 1994. Ian was recognized by the Industry as one of the individuals who significantly contributed to the ultimate success of the Microsoft Operating system. The award was presented by Bill Gates at Spring Comdex in 1994.
Ian received Bachelor's degrees in Mathematics and in Biology from the University of Rochester in 1982.

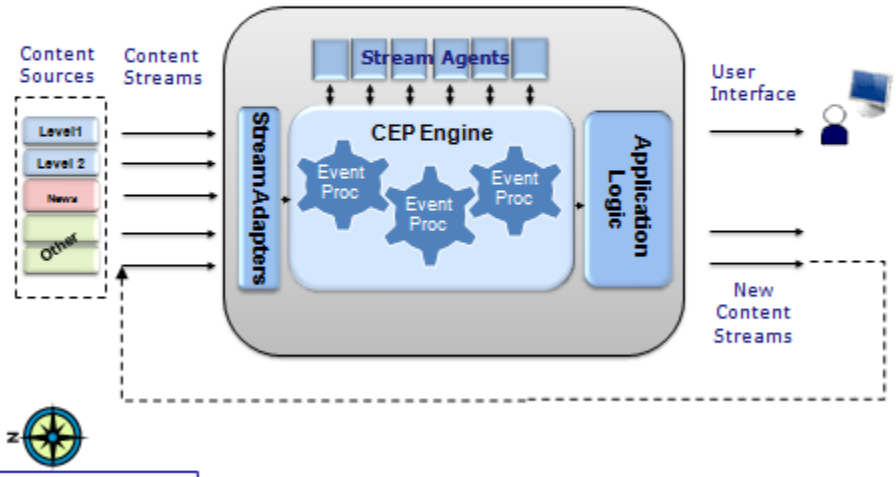
He reached at: ian.koenig@thomson.com.

Agenda

1. A framework for discussing Complex Event Processing
2. "Structured" News as a data source
3. The "fabric" for distributing content and its emerging capabilities for stream processing
4. Enabling new types of data sources creating new 'opportunities'
5. Hinting at a larger pattern for distributing content and the role that complex event processing will play.

In order to discuss the role of Complex Event Processing (CEP) and the interaction between the CEP engine that processes the event streams and the content distribution fabric that delivers the event streams, it is useful to have an architecture framework that helps to clarify the words we use. Following is the framework / model we use at Thomson Financial.

The CEP Framework model



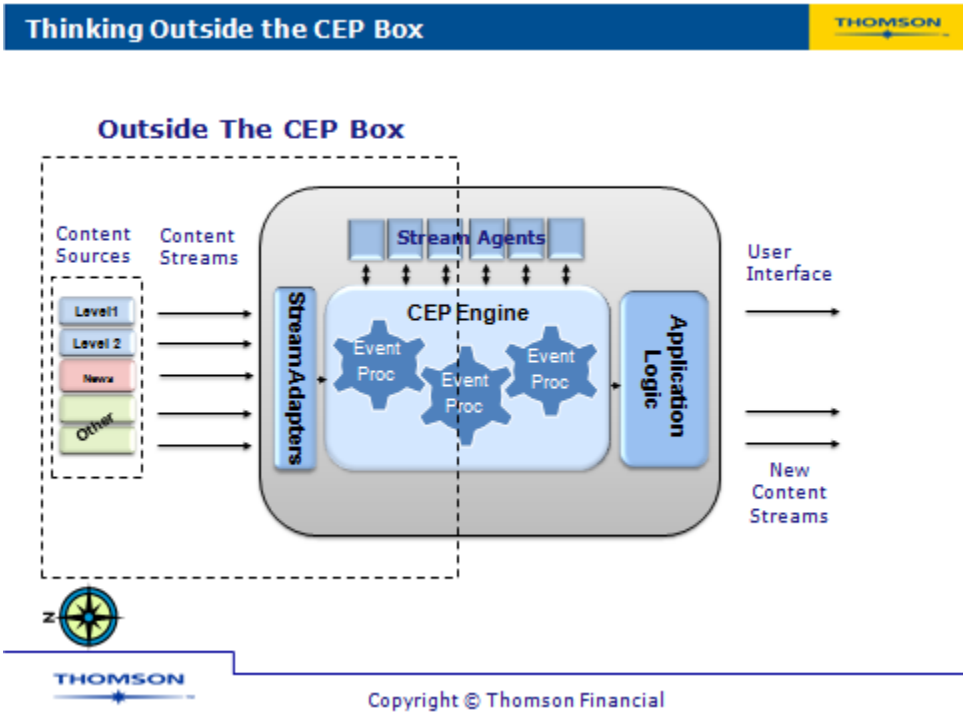
In this model, we have content sources that generate streams of data. How they do this and how these streams get into the engine is the subject of the rest of this presentation.

For the Content streams to be processed by the engine, they must be "adapted". We define these Stream adapters as the metadata / business logic required to describe the incoming content stream so that it can be processed by the CEP engine.

Stream agents are pieces of business logic that "stay in the box" and whose responsibility it is to "enrich" the data streams. They: (1) Listen to streams; (2) Use the functionality of the engine to correlate and analyze the streams either in time or against each other or against a temporal repository; And (3) they inject new streams or enrich exist streams. While this process is not dissimilar to what happens in the Application logic box, we find it useful to distinguish the enrichment process from the application process, even though they are both generally doing "complex event processing"

The Application Logic is then responsible for generating "what comes out". We find two major classes of this: (1) Activity monitors / alerts that are intended for a Human via a user interface and (2) New content streams that can be pipelined back into another engine for processing. For example, using the CEP engine to analyze market data, news and other input streams to perform Algorithmic trading is significant use case that we deal with. In this case the output of the engine (via the application logic) is an Order to buy or sell, generally encoded as a message in the FIX protocol that would either be input to an Order management / Execution management system or via Direct Market Access (DMA) to a trading venue.

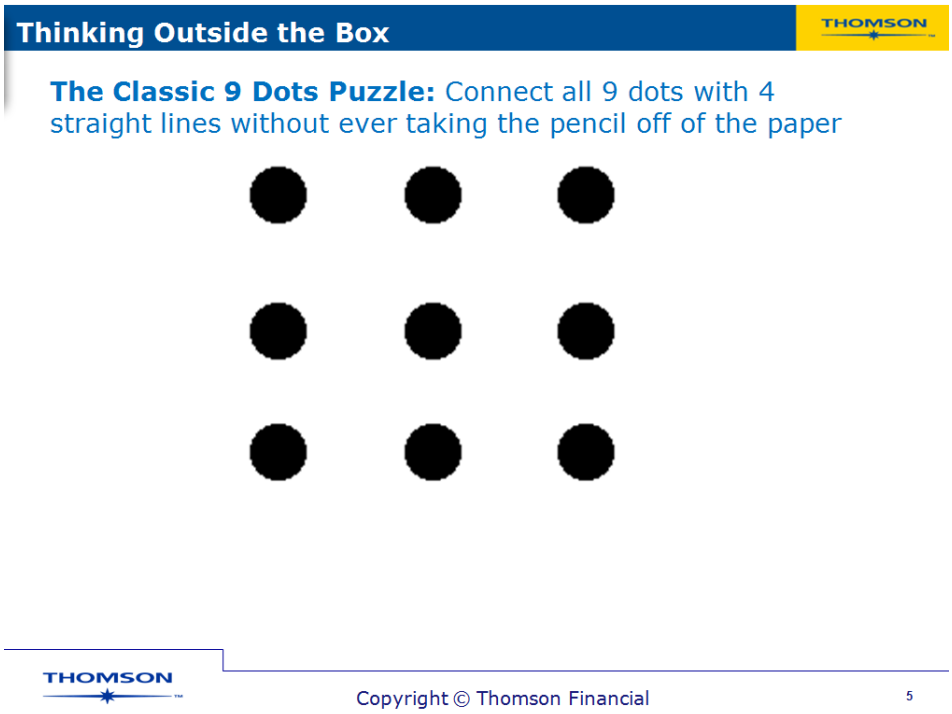
[Subsequent to the meetings, I realize that I have left out a major use case, which is the use of CEP engines to orchestrate Business processes, but I do not think that is a breaking change to the model].



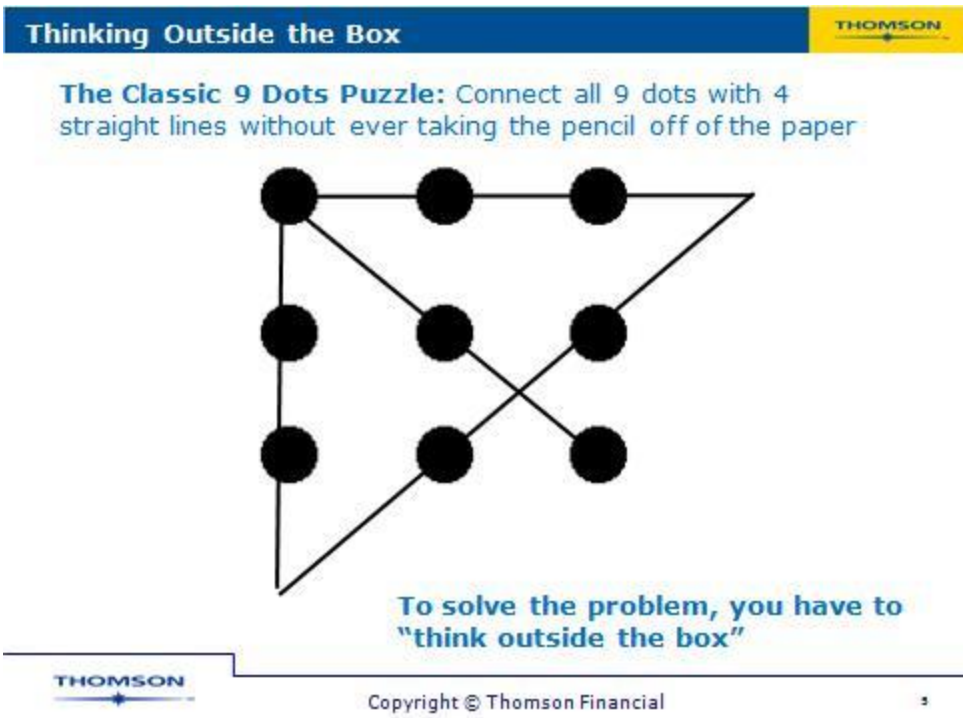
But as everybody else is mostly focusing on what's inside the CEP box, I thought it would be useful for this presentation to concentrate on what's "out of the box" or all the technology and the content

distribution fabric that gets the content streams into the CEP engine. I will also demonstrate the usefulness of complex event processing further upstream in the network and talk about how some of this type of logic becomes part of the distribution fabric / network and this adds value.

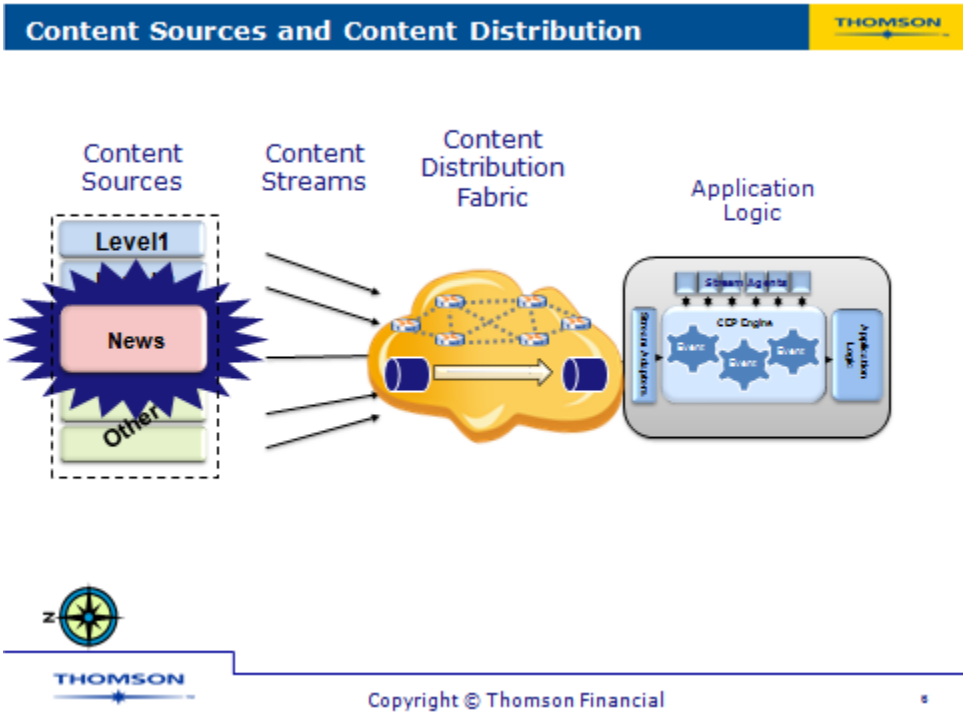
But before we get to this, as we are going to be "thinking outside the box", I thought it would be useful and potentially interesting to talk about where the colloquialism "thinking outside the box" came from. It's so common to use the phrase that it made me wonder what the derivation was. So I did a bit of research. And the nearest I could tell (using Wikipedia as a source), the derivation of the phrase comes from the classic "9 Dot Problem"



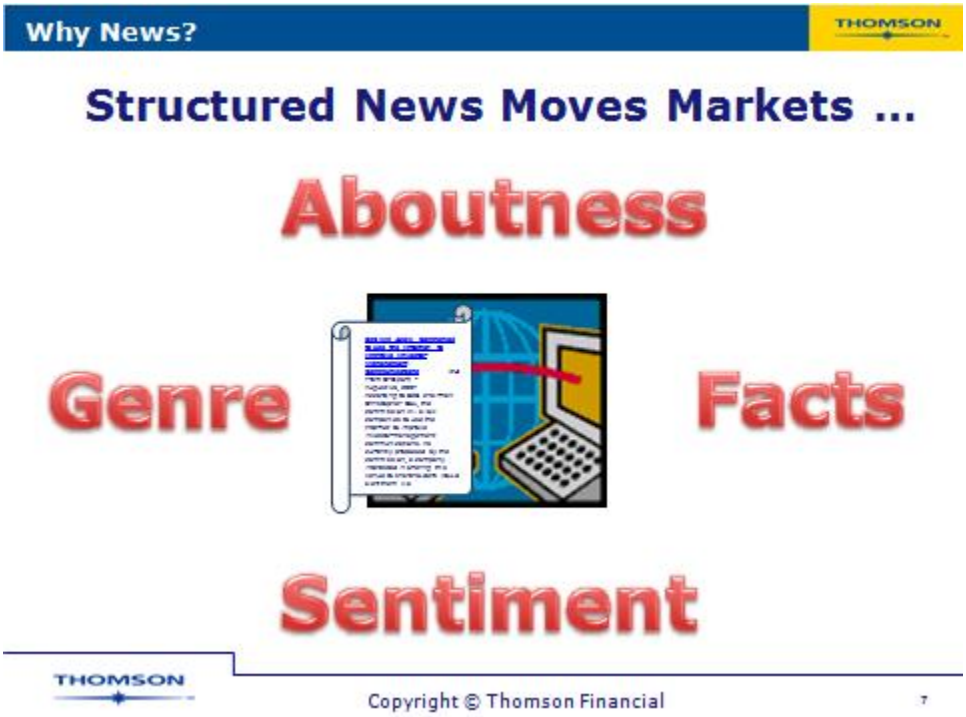
The "9 dot problem" poses the challenge of connecting all 9 dots with four straight lines, without ever lifting the pencil from the paper. And as we all know by now, the solution requires you to **"Think outside the box"**



Lots of focus exists on market data sources as inputs to CEP. By market data, we mean Level 1 and Level 2 data. Level 1 data generally means pricing data from trades and Bids/Asks from quotes plus volume and size. Level 2 generally adds "Depth of the Book" including all the orders at all the price points to the Level 1 data.



But as everybody else is talking about market data and in keeping with the "Thinking outside the Box", theme of this presentation, I thought I'd just skip over that and talk about the next content source of interest, which we think is: News. There is quite a bit of press about News as a new source of data driving algorithmic trading (pun intended)



So why is News interesting? Because News moves markets. But to use News as a source of algorithmic trading, given that News is textual in nature and targeted at Human consumption, we need to enrich the News content with computer processable constructs. We call this "**Structured News**" and the enrichment takes the form of specific XML tags that are imbued with computer processable information, embedded in the NewsML (XML) message.

We classify these tags into four general categories:

- **Aboutness:** What the news is about within a well-defined concept system (or ontology). Aboutness is defined by: **Entities** and **Subjects**. These are the: Companies, People, Markets, Places, Business Events, or Subjects (e.g. the Economy, Human Interest, etc.) that the news is "about". In database 'lingo' Entities and subjects are all Entities, but we differentiate those Entities that link other content sets (in our universe) together (like Companies, & people) and those that don't. Those that do are Entities. Those that do not are "Subjects". To further help distinguish these, often confusing terms; it is useful to note that Subjects tend to take the plural form. In other words they tend to be types of things and groups of things. Entities tend to take the singular form. An Entity is an Object. Examples always work better: Companies are entities because you think of a specific company as a thing. "Human Interest" is a subject, because it's a group of things not a thing. I wish I could be more precise in the language that describes these concepts because the distinction is important. But I am just not there yet.

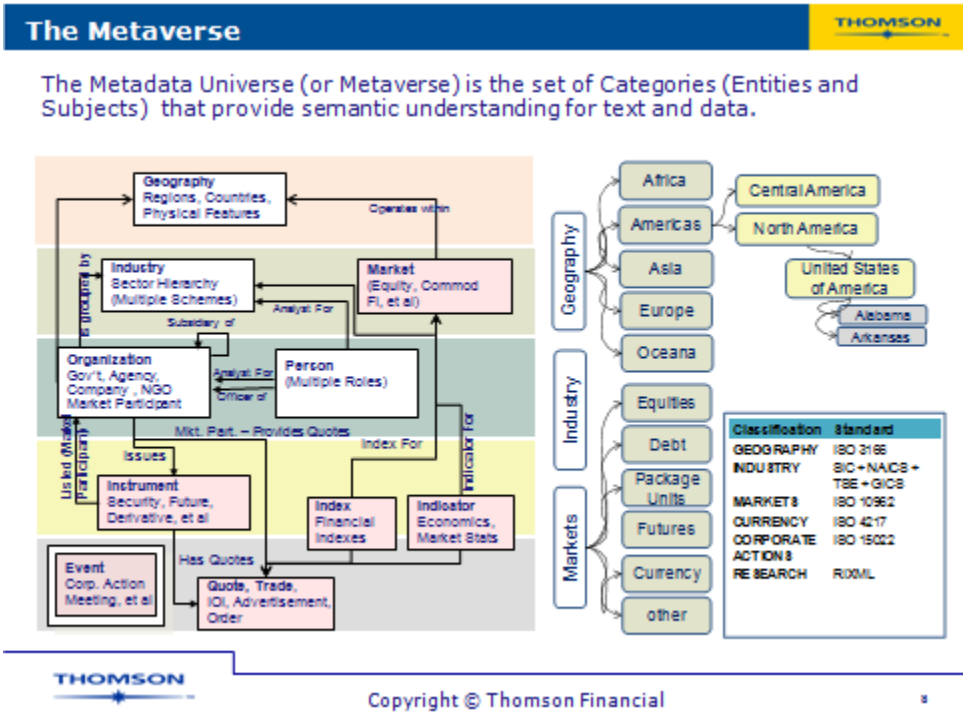
- **Genre:** (or Type): Is this a story, a feature, a market report, a blog, an opinion, a rumor, or something else?

- **Facts:** Specific numbers, such as Economic releases or Company Earnings that are tagged with XML elements
- **Sentiment:** Is the Story generally positive, negative or neutral about the subject(s).

When a person authors a research report, or offers an opinion or provides commentary, it's good to specify sentiment, because the author knows whether she has a positive, negative or neutral opinion on the subject. There is a bit of press about the technology for using computers to derive sentiment. We (at Thomson Financial) are somewhat dubious of the claims of these technologies, so we do not algorithmically derive sentiment. Instead we present the "facts" and let the consumer (probably using CEP engines or the equivalent) interpret whether the facts present a positive, negative or neutral position (unless of course we got sentiment from the author – which we treat as "fact")

Aboutness, Genre, Fact and Sentiment "tags" are added both at the document level and marked up in-line. For Aboutness and Facts, we use Computer algorithms to enhance the traditionally human process of categorizing news. This is called auto-categorization and I will talk in a bit about the really cool technology that does this.

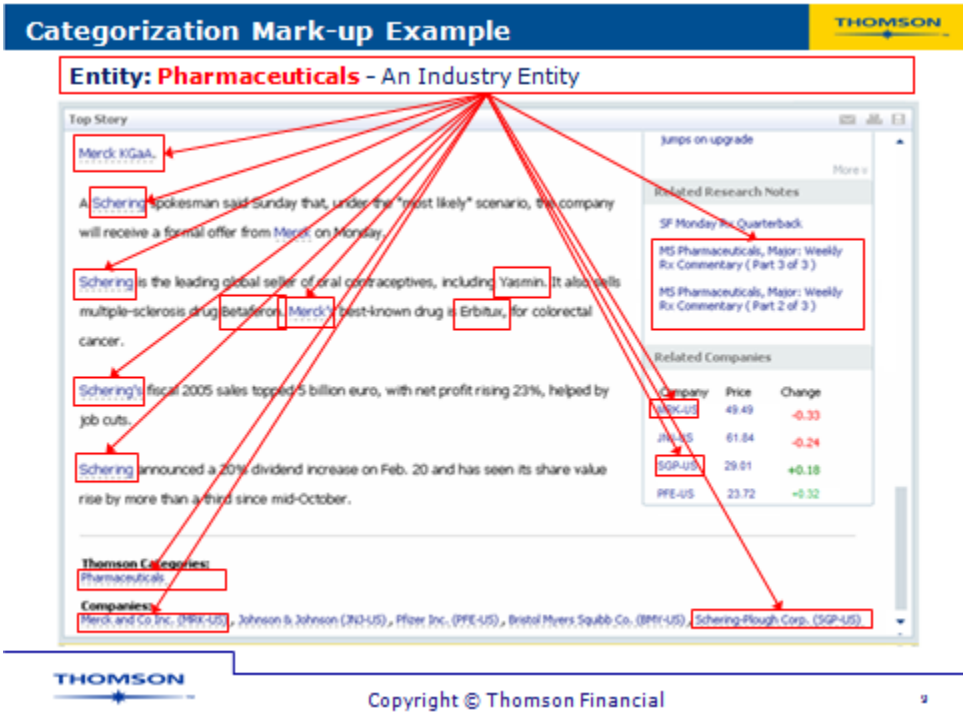
But first, let's dive a bit deeper into what we mean by: Aboutness



In order to know what a News story is about, you need a concept system that defines the universe of terms in your Aboutness vocabulary. We call this type of concept system The Metaverse (or metadata universe). For Thomson financial, the Metaverse is called "Thomson Master Categories (TMC)". In Thomson Master Categories, the concept system is presented as an ontology of terms, the backbone of which are the Entities and Subjects I spoke about before.

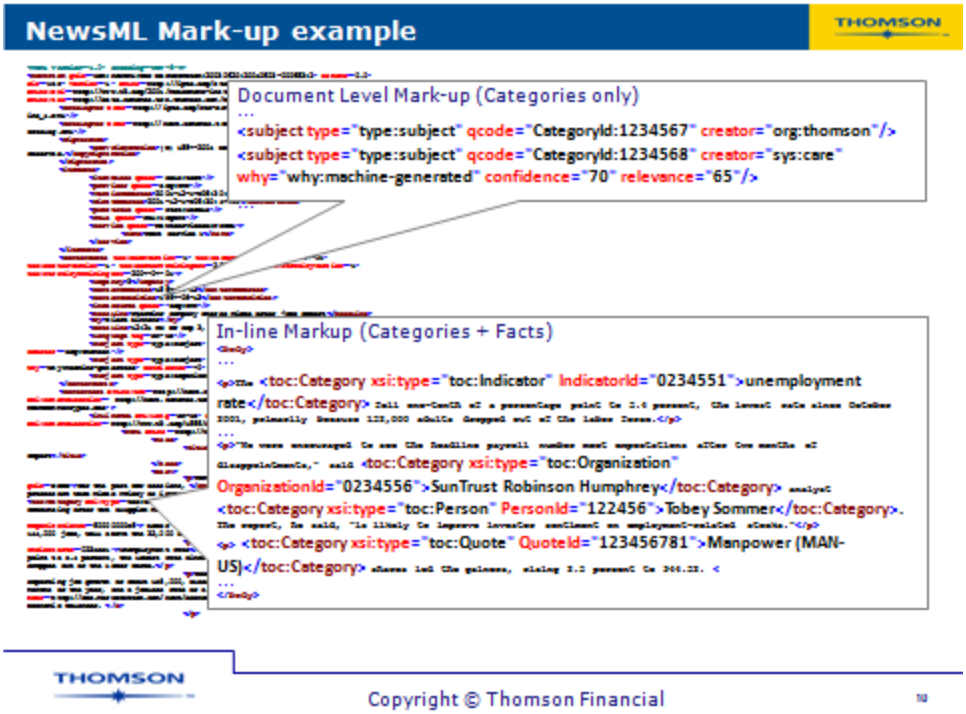
The canonical ontology is intended for computers. It consists of the set of tags that we attach to the documents. In the canonical ontology, the philosophy is "the more precise and the more granular, the better". Computers can easily deal with ontologies of 10's of thousands of terms.

Let's take a look at how a Human interface would use the enriched News content to add contextual links. It should not be too difficult to imagine how a computer would use these same techniques to algorithmically trade.



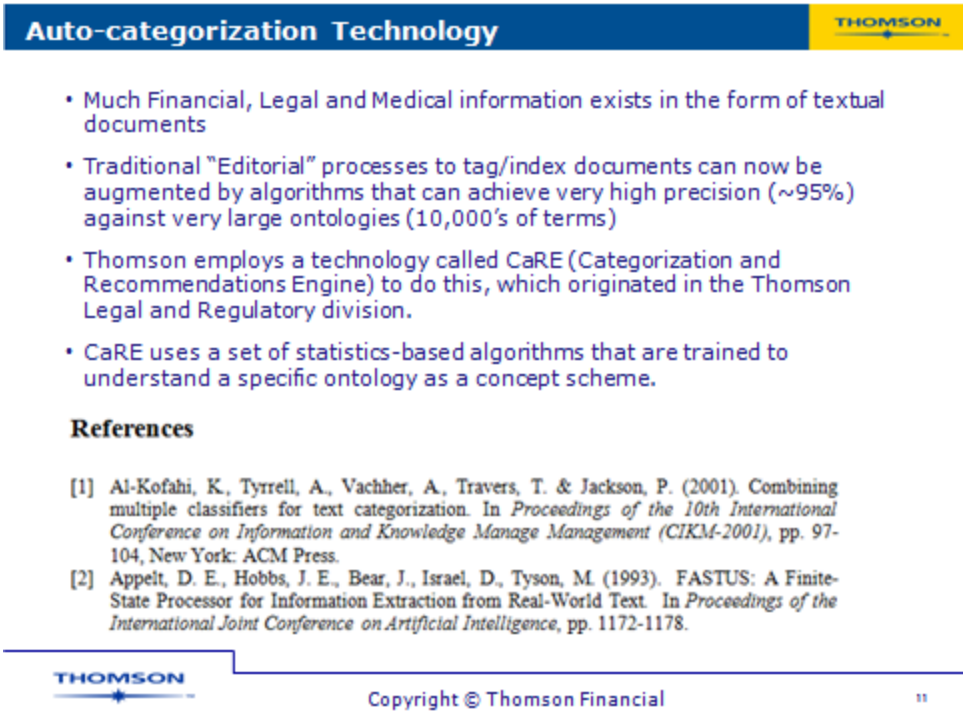
In this example, we have enriched the News document with Aboutness information. We know, for example, what Companies are referenced. In the above example, which is about Merck and Schering Plough and the Pharmaceuticals Industry, we can use this contextual information to present the latest prices for these companies and related information on the Industry (such as related news or a sector based index). But when we say something like "What's the price of Merck", we are saying a mouthful; because we really don't mean the price of the company. We mean the latest quote from the default pricing venue (I.e. a Stock Exchange, like NYSE) for the common stock issued by the company. So while a human immediately know what you meant when you asked for the price of the company, a computer has to navigate the entity relationships in the Metaverse to figure it out.

For the "geekier" among us, let's take a look at the actual NewsML, and how we enrich it to embed computer processable information in what is otherwise text targeted at a human.



NewsML is an Industry standard, XML schema used for marking up and distributing News. NewsML G2 is the most recent proposed version of NewsML and is the version we refer to here. When we enrich the News, we embed tags that identify the subjects and Entities that the News is about. When we find a relevant fact, we mark that up too. There are two levels of tagging: Document level and in-line. The document level tagging tells you what the whole document is about and when we find an Entity or a fact in-line that we recognize, we put an XML tag around it to identify it in situ. So we can pull out a company reference, or a reference to a person, as in the example above. When we find an entity, we match it against an authority file that gives it a GUID, which is also permanent (i.e. never changes).

Marking up XML is not something that humans are particularly good at. Luckily there are computer algorithms that can assist.

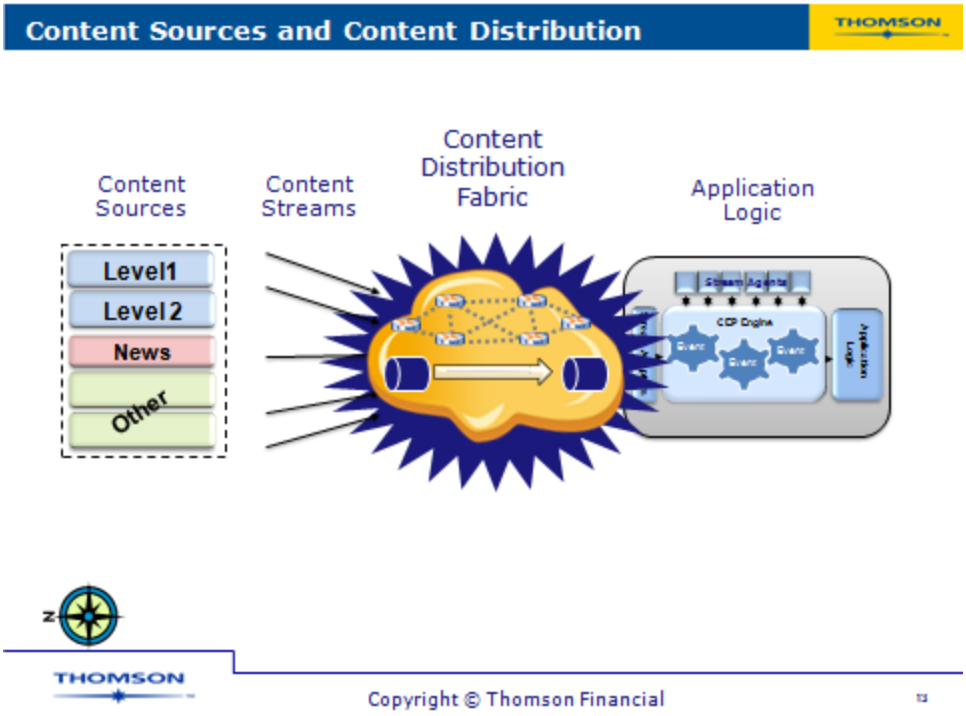


The technology we use at Thomson is called: CaRE (Categorization and Recommendations Engine). It is based on a framework and a whole bunch of science around information processing. (Please see extracted references in the slide). For identifying subjects, the technology uses statistically based algorithms that you train. In other words, you show the system for each subject in your concept system, maybe 10 documents that are about the subject and maybe 5 that are not. The technology framework figures out which combinations of words and phrases are likely to "mean" that subject. If you don't like the precision you get, you add more training documents. The structure of the document is not particularly relevant to CaRE. Care treats the document as a "bag of words".

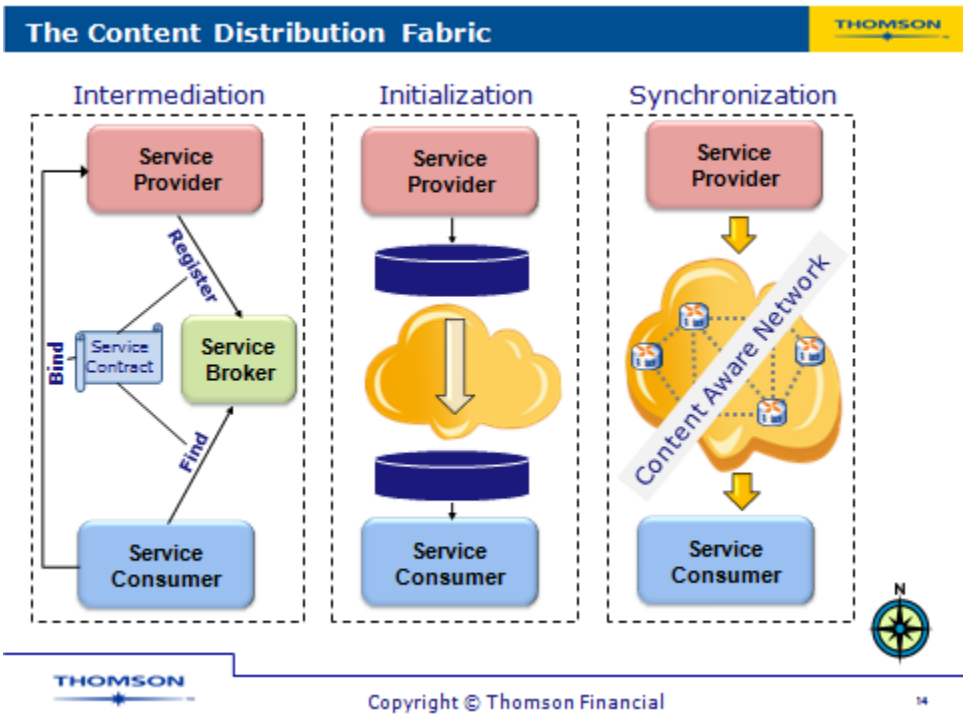
Entity Extraction is a bit more algorithmic (as opposed to being statistically based) and a bit more dependent on understanding the structure of the documents. Entity extraction is done in two steps: identification and resolution. The Identification process finds word groups that "look like" a company or that "look like" a person (or another Entity) . It also tries to pull out secondary evidence, like the Industry of company or the role of a person. The extracted entity and the secondary evidence is then passed to a Resolution process, which matches the entity name and the secondary evidence against an authority file, that would assign the Permanent GUID if the Entity is known, or initiate a process whereby a Data analyst may choose to add the new entity to the authority file.

Now that we have talked about News (noting that it is not a number-oriented row/column based data source) as an additional "interesting" content stream that we want to process and correlate with Market data other data streams, let's talk about the nature of the distribution fabric (or network) that exists to distribute content to the consuming applications.

We have market data content streams and now we have news content streams. We know about the complex event processing logic that consumes these streams. Now, we will cover the Event processing logic that we can build into the distribution fabric between the source and the destination.



In order to understand the nature of the Content Distribution Fabric, we will discuss it in the context of an architecture framework / model that helps define the words we use in the context that we use them.



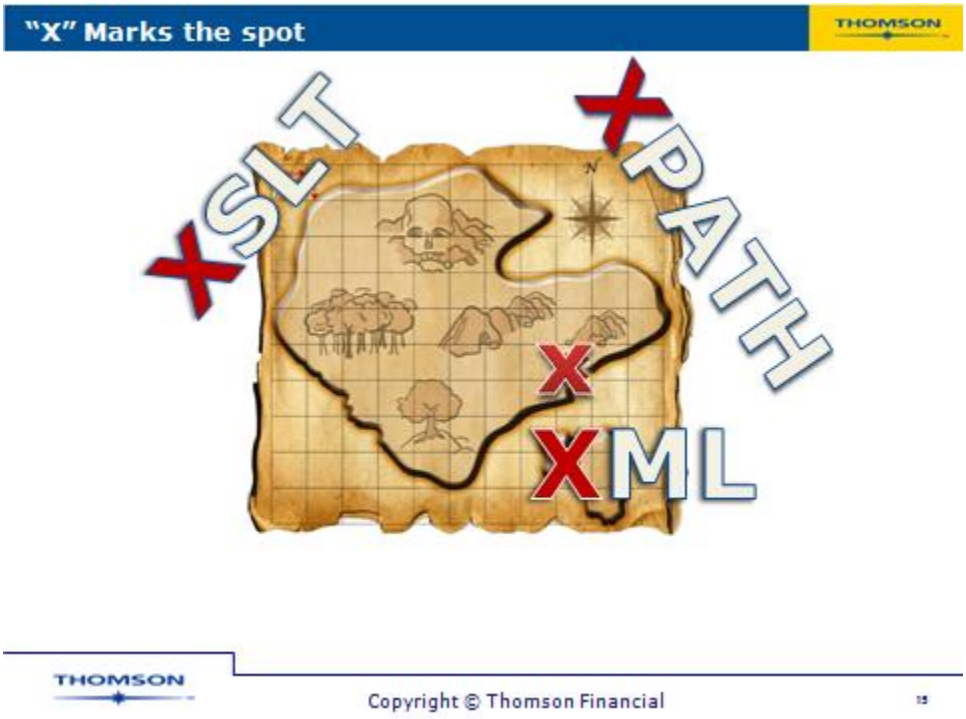
The Fabric itself provides three main capabilities:

- **Intermediation** – The process by which we connect service providers (e.g. content sources) and service consumers (e.g. CEP applications). These must be "loosely coupled" so we provide a

meta-data driven service broker to connect them. By loose coupling we mean:
the ability to make a breaking change in the interface (i.e. non-backwards compatible change) without actually breaking the application, so that we can move from the previous version to the next version upgrading provider and consumer independently, one at a time, and in either order. The Service broker follows a "Register, Find and Bind" pattern. Providers publish onto the bus. Consumers subscribe. The bus mediates.

- **Initialization** – is the process by which we get a service consume starting from scratch with an empty database to a known "synchronization checkpoint" as defined by the content source.

- **Synchronization** – Is the process by which we keep the service consumer updated as changes are made to the content source. This is event based distribution of messages over a content aware network. The content aware network is based on a set of "XML routers" that basically provide a pub/sub network, where XPATH expressions are used to define "interest" in a steam of content, and the XML routers then pass XML documents between each other in a manner similar to how a standard routers use IP Addresses to know which packets go where.



XML is the "Enabling technology" for content based distribution. There is a ton of industry effort targeted towards XML and being able to leverage that effort pays dividends. The key XML standards (for us, today) are:

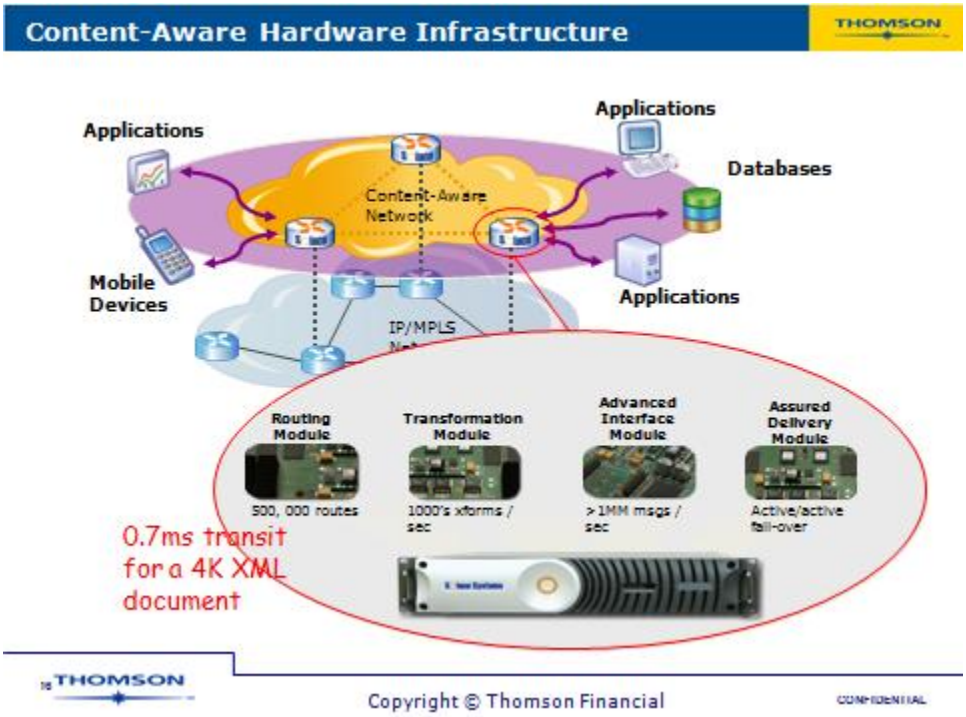
XPATH - The XML Path Language (XPath) is an expression language that allows for addressing parts on an XML document. XPath models an XML document as a tree in which elements, attributes, and content are nodes. Its name is derived from the use of the slash separator to descend the branches of the tree. An XPath expression "selects" a node-set (one or more nodes), e.g. a string, a number, or a Boolean.

XSLT - XSLT is typically used to transform an XML document into another XML document. It does this with style sheets. XSLT stylesheets are similar to Cascading Style Sheets, with two exceptions: XSLT

stylesheets are written in XML, and XPath is used to determine which parts of the original document to transform.

An interesting standard that may be interesting as well is **XQUERY**. Today's CEP engines seem to use extensions of SQL as their language (not all, but many). But if you accept the proposition that non-numeric, non-row/column oriented data becomes important for CEP engines to process, then one has to ask the question whether we should jump to XQUERY or keep extending SQL. I do not have an opinion on the subject (at least not today).

The major impediment to the wholesale adoption of XML for encoding data (at least for us) has always been the technical limitations associated with the size of the message, the performance impact of parsing the message and the cost impact on the distribution networks. But there is a new class of technology that is XML aware, routing content by interest, where the XML logic is implemented on hardware, so in addition to being functional it is also fast.



The technology we have started to use is the Solace Systems VRS/32. Its not the only one out there, but it's the one we have settled on. And with this technology as the basis of our content aware pub/sub distribution fabric, we can achieve:

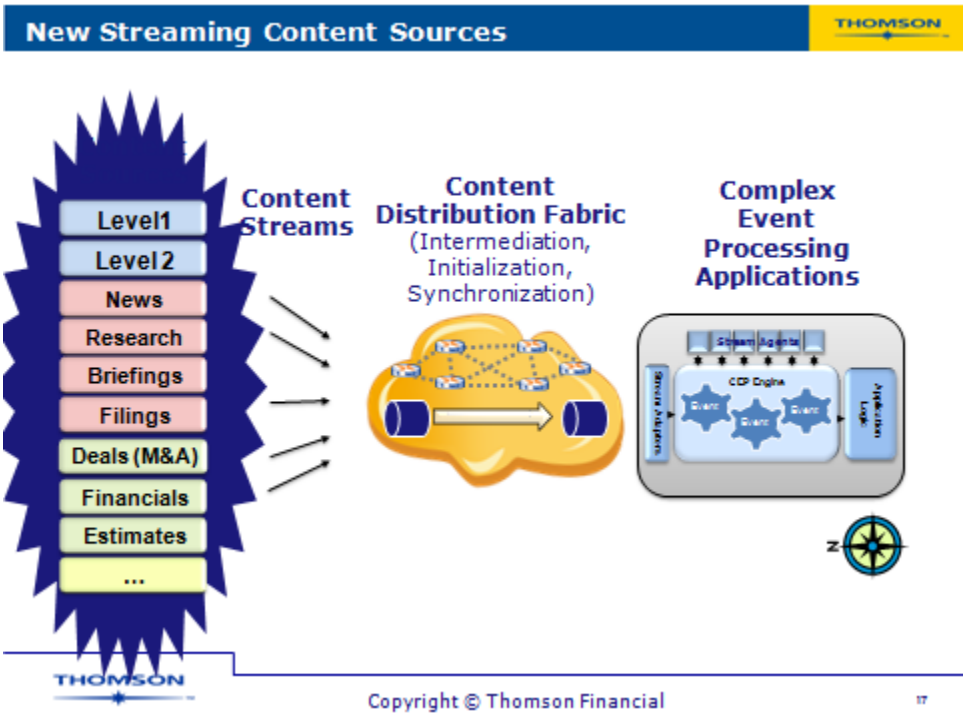
- Message throughput (across an individual device) of > 1 million (small) messages per second.
- Thousands of XSLT transforms per second
- Active/active fail-over without message loss (in a single data center)
- And most importantly, message transit times (across a single device) of 700 microseconds for a 4K XML message, under load.

So effectively, by using hardware based content routers, XML should no longer be a technological hurdle. I need to qualify that last statement a bit. We still do not envision distributing market data via XML any time soon. Given data rates and explosive growth, this just does not make sense at the present

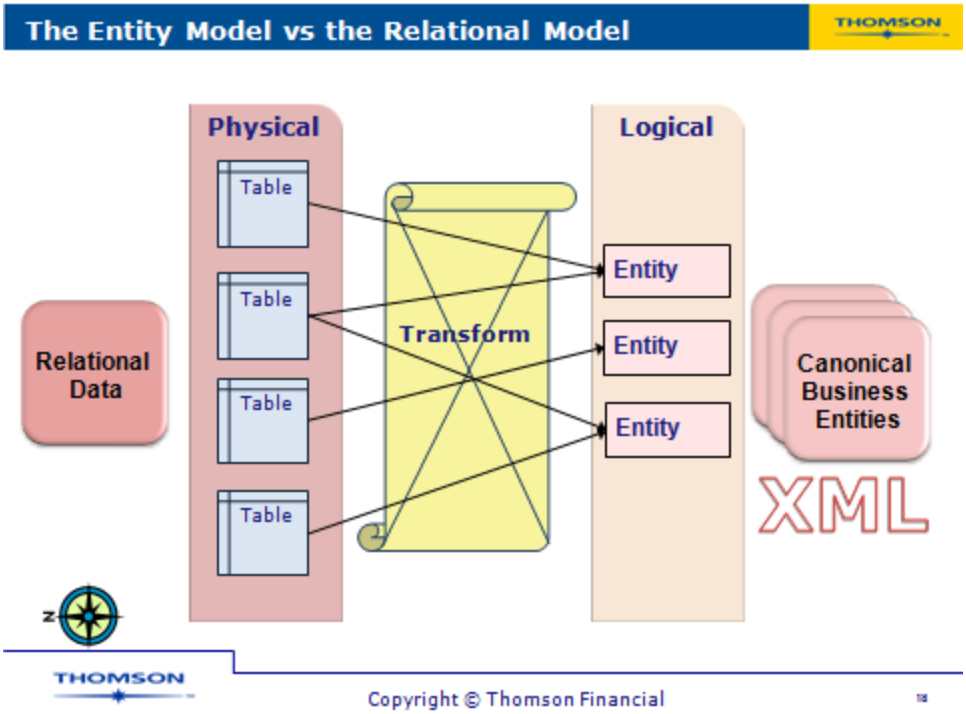
time with the present technology base. But for lots and lots of other data sets, it makes perfect sense and the benefits are huge.

For example, as a subscriber you can get the network to filter the message streams based on any element or attribute in the XML. You don't have to "design the namespace" of available topics that can be subscribed to. Once XML is published onto the bus, a subscriber can retrieve it, by specifying an appropriate XPATH expression.

The network thus allows publishers to publish in a subscriber agnostic way. Since the network does the filtering, the subscribers are not overloaded when additional content becomes available. Another source of network optimization is that the subscriber can define an XSLT transform to "reduce" the message and only pass that subset of the schema which is of interest. So if the publisher creates a schema with 4000 elements and the subscriber only wants six, an XSLT can be defined to reduce the message down to the six elements of interest. This reduction is performed only for that subscriber. A different subscriber can define a completely different XSLT transformation / reduction, or just take the full message.



So now that we have talked about News as a content stream and we have talked about the content distribution fabric that provides some interesting capabilities for distributing to subscribing applications, such as CEP engines, we should immediately see that the possibilities for other interesting sources of content that can be used by algorithmic trading applications. The possibilities are numerous. Content such as: Broker Research or Estimates, Company briefings or flings, Deals (mergers and acquisitions) and more can all be made available in a form convenient to process, analyze and correlate with other content streams. Each of these types of data would be transformed into an XML model and events would be streamed over the distribution fabric to subscribing applications.



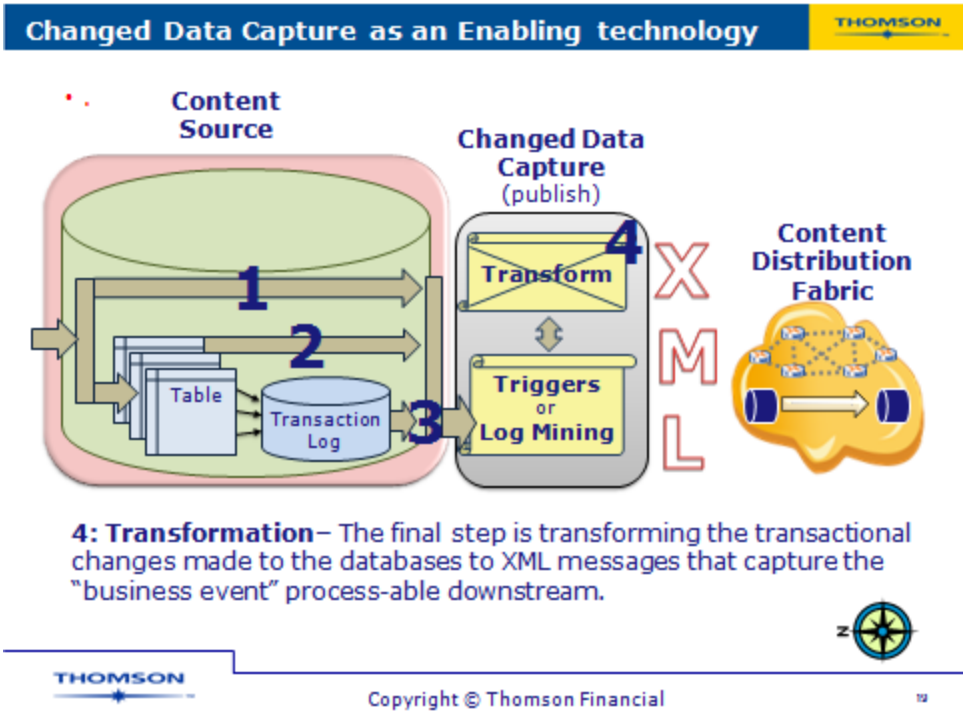
Now that we have passed the "XML hurdle", the next technology issue we hit is the act of creating the proper XML data model for distribution. Most of our data is stored in relational databases and relational databases store data in physical tables. To create truth, the Relational data model is often highly normalized, so a single data element exists exactly once. There tend to be lots of tables, lots of foreign keys and the databases are optimized for update rather than retrieval.

But what we want to come out of the database in the form of events are logical constructs we call canonical business entities. Furthermore we want the granularity of the event stream these to map to the business events that caused the database to change in the first place.

Unfortunately by the time you want to create the output, in many of the existing content masters, the incoming business event has been totally deconstructed in order to store the data physically in the table structure. What we really want to do is to reconstruct that event and transform the physical table based model into the logical canonical model.

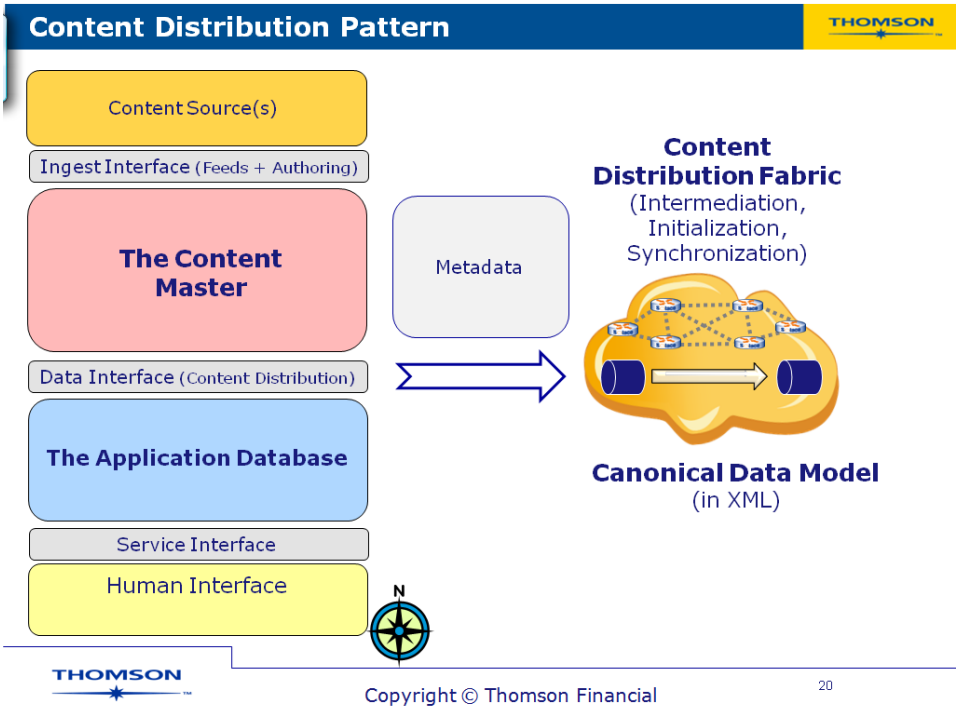
The newer databases we have created use an architectural model, we call a "publishing pipeline" where the database business logic preserves the business event (as it came in) and uses that to guide the production of the output. Older databases, where the deconstruction of the originating business event is complete, just have to do the heavy lifting to reconstruct the logical data model by extracting and transforming the physical table model.

Fortunately, many of the databases we have looked at nearly preserve the business events in the form of the transactions that are committed to the database. In other words, whether by good planning or by good luck, the database transactions as written map close enough to the business events we want to model to be very, very useful.



But you still have to extract the data and process the events and emit the messages and for many databases (and database developers) that's hard. Most of these databases were built assuming the data would be "at rest", not "in motion". In fact the whole concept of event processing and "Data in motion" is foreign to many database developers. But once again there is technology that can help. This class of technology is called **Changed Data Capture**. It either uses triggers to recognize that data has changed or a process called log mining. Most modern databases have a transaction log, which is the golden source of all transactions committed to the database. In fact you can look at the database itself as a "latest value" representation of the transaction log (okay, that's a bit abstract, and not important to the discussion – but still interesting). Log mining is a process whereby Changed Data Capture technology watches and recognizes when a transaction has been written to the transaction log (meaning the transaction has been committed to the database). It can then read the transaction, perform the transformation (which of course you have to define) and initiate the message flow downstream. This could be via its own middleware to a set of target applications, or onto your middleware bus via an API (like JMS).

While it is best for a database that is mastering data to use the "publishing pipeline" pattern where the incoming business events are preserved and messages are emitted once the transaction is committed, for those not built this way, Changed Data Capture technology can make the act of moving into the event-based world a bit easier.



This all hints at a larger architectural pattern governing the flow of content from source to consuming application, and content distribution systems in general.

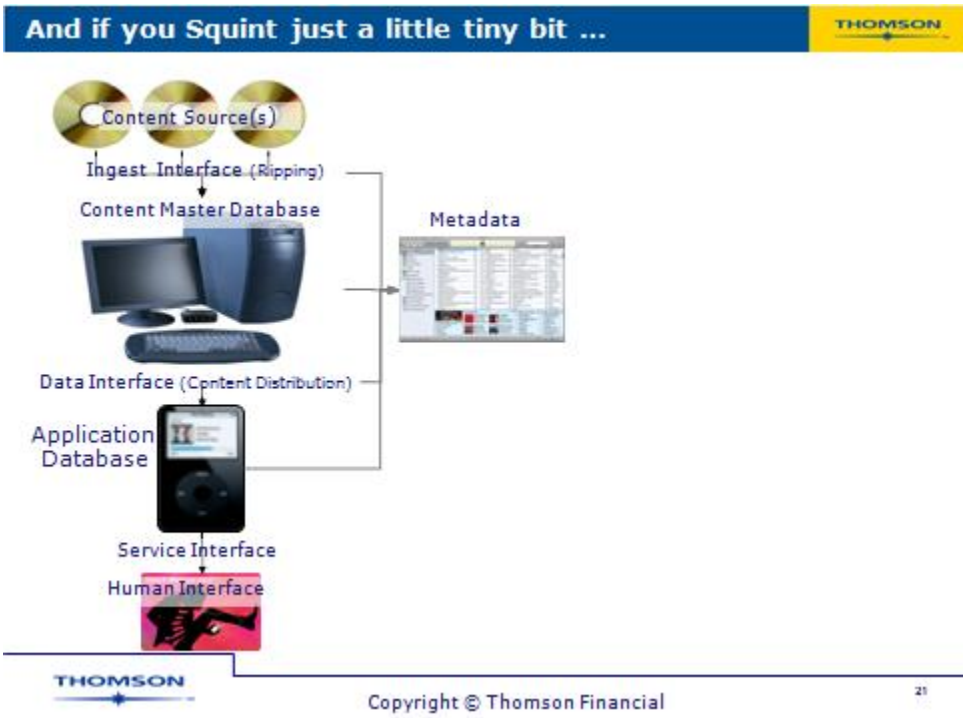
I've read lots of the literature and research that refers to the Enterprise Database as this amorphous "mass" of uncoordinated data and which positions SOA (Service Oriented Architectures) as the white night to slay the dragon of chaos. Maybe it is? Or maybe, with any technique like SOA or OO or MDM or "pick your favorite acronym" comes a degree of architectural rigor that if properly applied both technologically and organizationally, creates order from chaos? Either way, the main value of Enterprise Architecture to any business is to provide a framework in which business imperatives can be translated quickly, easily and with high fidelity into technology solutions.

Our **Content Distribution Pattern** learns the lessons from market data and data-feeds and applies those lessons widely across the world of content. What we learn is that The Enterprise Database is not a single thing. In fact we see two distinct classes of databases (and we use the term database very loosely here):

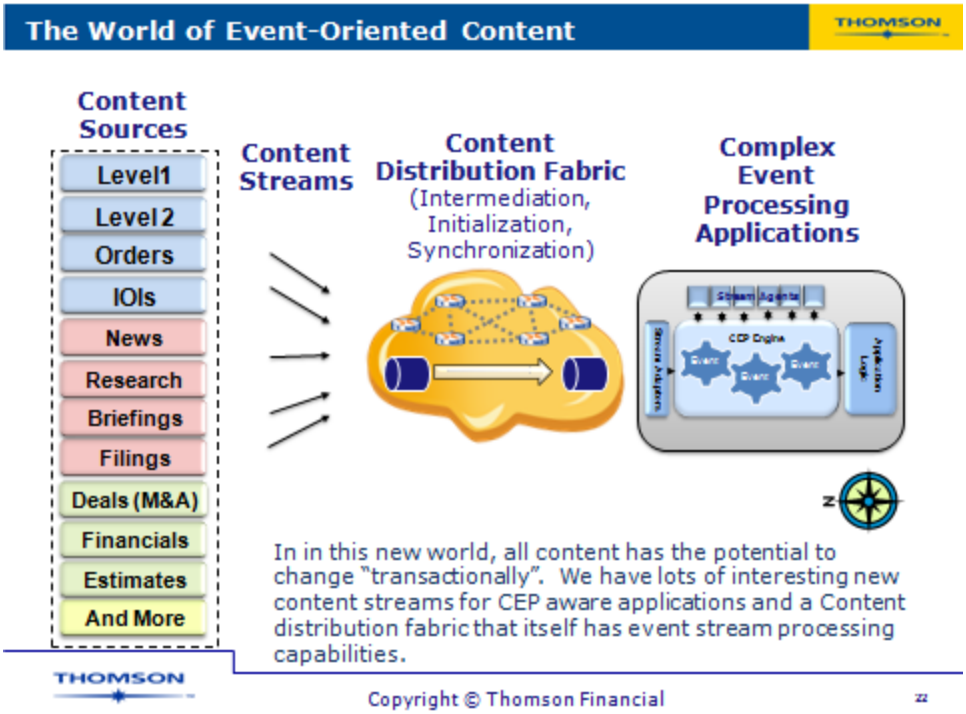
- Content masters – which exist to house the single version of the "truth". These are databases of record or the house of "gold" data.
- Application databases – which exist to make predictable queries, fast. Here is where use the term "database" very loosely. Often these are databases. For content, almost always. For high volume, high velocity data, these are purpose built, memory based databases. And this is the place in this pattern where CEP engines are making their initial impact.

Content masters are generally highly normalized, highly optimized for update, and never delete data (at least those that follow the rules are). Application databases are generally highly de-normalized and optimized for search and retrieval. Between them is the "Data Interface", which provides: Intermediation, Initialization and Synchronization capabilities. The Content Distribution Fabric provides the concrete implementation for the data interface over a Content aware network.

This pattern may seem a bit abstract, but if you squint just a little, tiny bit, you may notice that you've seen it before…



If you have an IPod, you've done this. You've followed the Content Distribution Pattern. Because you've ingested content from its source; you've created a master database (on your PC or Mac); you've added metadata (via ITunes) to properly describe the data that you've mastered; and you've distributed that data (either all or in subsets), to your application database (the IPod itself) which holds a copy of the data in a database optimized for retrieval, and accessed that data via the Human Interface. So, maybe its not so abstract after all?



And in summary, while today we really do have this dichotomy between data in motion (like Market Data and News) and data at rest (like stuff you search for), as we move forward, the line between these will change (but not go away). More data can and should be dealt with "in motion" and handled via event processing technology.

After all, we are all Event processing animals. We handle events as they happen, like when a phone rings, we answer it; when the alarm clock goes off, we wake up. I don't know anybody who sits there and stares at the alarm clock all night until 6:00 AM, in order to realize it's time to wake up. We process the "clock ringing" event. But for some reason, most of our applications are written to watch the clock, not process the event. Hmmm?

There is some interesting technology coming to market that breaks down some of the traditional technological barriers that have made processing events hard. I've described a few in the context of this presentation, but there are more out there, and I am sure, even more on the drawing board.

As these technologies make their way into the enterprise infrastructure, we will be moving down the road of event based distribution of information and complex event processing in the application tier.

I hope that by "thinking outside the box", I've been able to offer some interesting and relevant information that provides context to the kinds of exciting things we can do with the technology we have today. I hope it also provides some insight to the vendors of these technologies as to where innovations in the technology landscape will provide value in the near future.