Short Extracts from Part 3 and Part 4

+Part 3: Software Semantic Evolution with SOA, Microservices, RAML, DataSense by MuleSoft and the next step
+Part 4: Big Data and Semantic Tools at work

Part 1: Knowledge Driven Architecture | **Part 2: Transitioning to Semantic Cloud**

Part 5: Semantic Toolbox and its Magic for Validation of Development

The message from 2040 | **Discussions with the first readers** | **Buy the book**
If you like the magic of web and mobile development and would like to become a magician

---

Based on events of 2040: The second law of thermodynamics in economy

Things did not happen overnight. In ten years after the book publication, some of us were getting ready to celebrate "the age of stability"…

I remember those peaceful days, weeks, months, and even years. But then some gradual changes in the economy became visible. Economy was slowing down, decreasing demands on products and services. The market reacted quickly and the corporate world started shrinking. First it cut back on R&D and then its overall employment budget.

The best explanation I found in the press was a long article titled "The second law of thermodynamics in the economy." Simply, the law states that entropy in any system is always increasing unless special efforts are taken to prevent this natural tendency to disobedience, disorder, and chaos. Apparently, the recent efforts of our workforce were not enough to keep healthy economy. In a convincing chapter "Victory means risk," the author of the article connected the level of efforts and their results with the level of risk taken by the companies and individuals. "It turns out that the recent corporate changes have reduced our motivation to individual risk and individual achievements. The economic interpretation of our so called "stability" is stagnation." I did not finish reading the long list of examples provided in the

article. The idea was clear. Limiting deviations from the average course, via governmental control, corporate "collaborative bureaucracy," we consistently re-created short-term stability inevitably followed by stagnation.

New corporate management did not follow the crazy practice of mass layoffs at lunch on Fridays. Instead, acting in a collaborative manner, most of the companies offered the volunteering "cut your paycheck" options. It was probably the first time, when not occasional folks but all of society took the hit, making the hit not so painful for individuals. It was an interesting time when we discussed the positive and negative consequences of the new management mentality. We've learned a lot since those days.

I do not remember who first suggested "My Risky Deal Offer". This could be a project or a business move offered by an individual or a group. The offer would have business details including investments by a caller and the match expected from a company. This was like a collaborative startup, where a caller would provide a significant contribution and a company would match some negotiated percentage to support the project. Our stability was disrupted by the avalanche of these "collaborative startups". They worked days and nights and still most of them failed. But survivors brought great results and pushed economy back on track.

The modeling factories in the Sahara desert were among the best achievements of those days. Designed to scale, the factories consistently increased production for many years. The recent negative results and the report submitted by an expert in robotic physiology (also a robot) were completely unexpected.

The report stated flatly that growing production required more parallel processing. New features demanded by clients required more knowledge exchange between the robot groups, which in its own turn required more parallel processing. Each robot dynamically acquired as many processors as needed. There was no shortage in computer power. But massive parallel processing created an enormous amount of problems. Complexity of data synchronization, networking, multithreading and other expected and unexpected factors grew exponentially.

The biggest problem was not really technical, but more related to robot psychology.

People can ignore extra data. Specially trained people can ignore multiple disturbing factors. It does not mean they make good decisions, but they continue functioning at some level. Robots were designed for optimization. They have no motivation to limit their data flow by the "need to know" or other artificial rules. While people often tend to control information as their key to power and individual success, robots try to support other robots providing all information for cross-examination from multiple points of view. Working simultaneously with many knowledge domains opened new opportunities, but also created new dependencies, increasing the decision cycles and required resources.

The report predicted that modeling factory production will continue slowing down until they reach some critical point that we passed several months ago. This will result in a violation of the agreement between the company and the clients. This might be the end of the company...

Read more in the book…

Part 3: Software Semantic Evolution with SOA, Microservices, RAML, DataSense by MuleSoft and the next step

Good old times of programming "all-in-one"…

Do you still remember good old times when a programming code included hardware drivers, data management, and business logic, - all together? We would call it spaghetti today, but at that time this was the only way to make it work. From zeros and ones we moved to assembly language, the first step in a semantic evolution of art of programming. And then software started its ascent over the ladder of architecture layers.

Architecture layers

Operating system developers, such as Sun Microsystems (currently Oracle), Microsoft, Apple, and several more took care of the system layer or more previse – operating system layer. Database vendors, such as Oracle, Sybase, Microsoft, and several more took care of the database layer. Most of programmers became application programmers, who built the application layer on the top of the giant shoulders mentioned before.

Application monsters

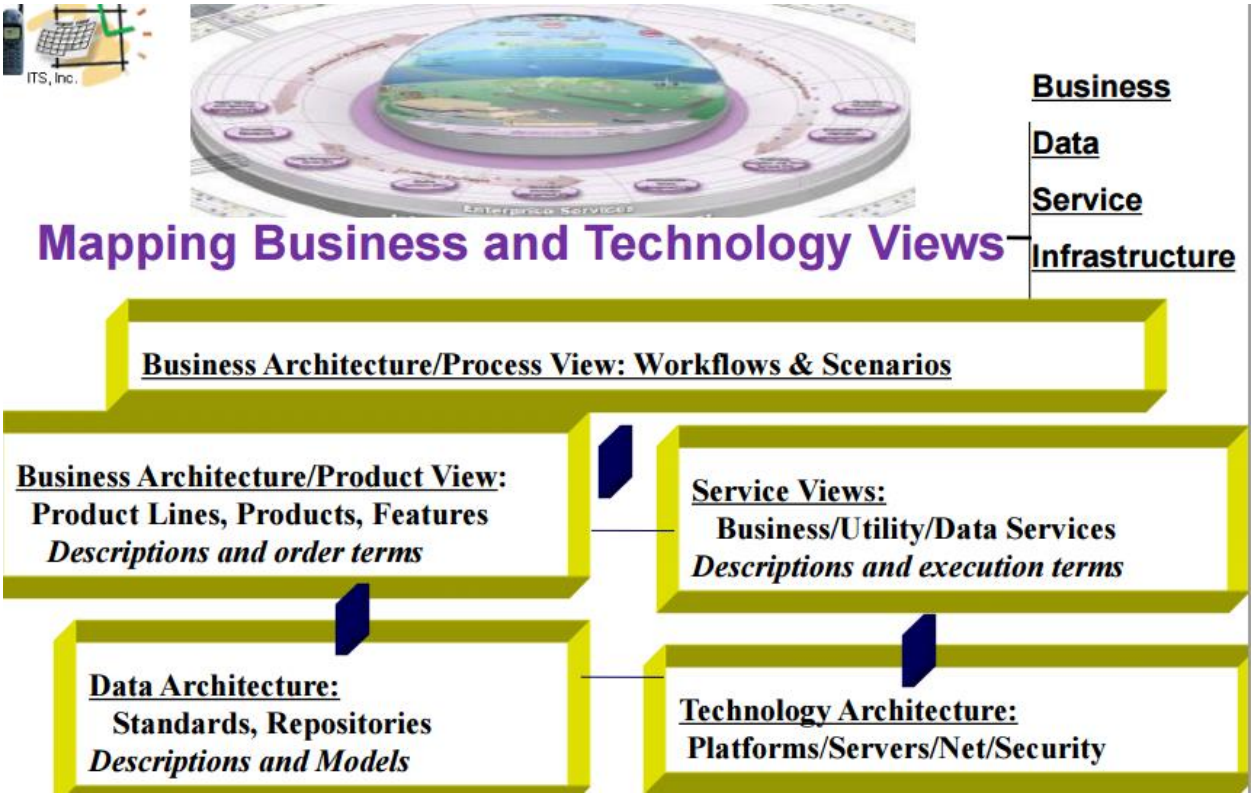
Divided by corporate barriers and working under "time-to-market" pressure, we replicated data and application functions and produced software that is neither soft nor friendly, lacks flexibility and teamwork skill, and is barely ready for integration into new environments. By producing "more of the same" we actually increased entropy and slow down the pace of technology [1].
Long projects and inflexible, fast-aging applications (that cost a fortune to maintain!) created more pressure for a better Business - Technology Convergence. Developed in isolated departments, applications often duplicate business functions and, with their growing number of features, become unmanageable and unpredictably expensive monsters.
Business changed their appeal to IT and development – it is too slow.

It takes multiple layers and teams to translate business requirements into Boolean Logic and bake it together with many old and new functions. The resulting cake is too firm in spite of its name – Software.

Service-oriented architecture (SOA)

SOA is a software architecture style that focuses on service components (services) that are reusable across multiple applications and enterprises. While Service-Oriented Architecture (SOA) is an old concept, current standards and technologies have paved the way to add efficiency and gain strategic advantages for the enterprise to quickly introduce new, or change existing business features.

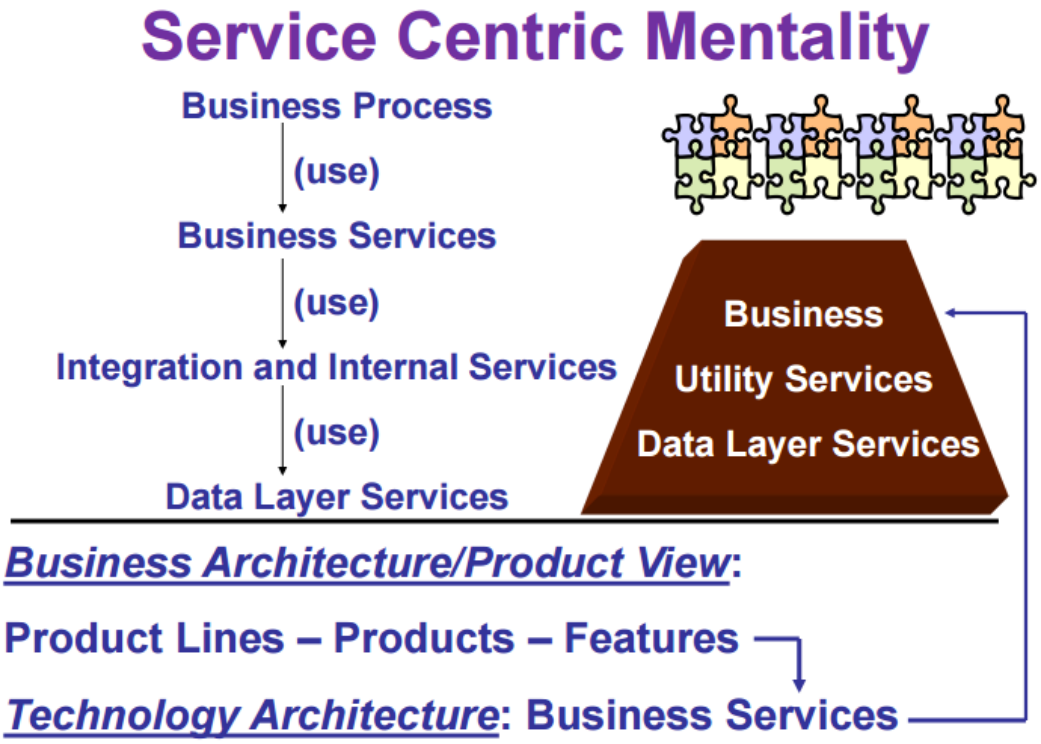
SOA helped translation of business products and services into architecture artifacts, starting from Business and Product Architecture Views and following with the Service Views, then Data and Infrastructure
Architectures.



SOA promised to simplify the transition from business vision to software development. This promise is not yet fulfilled. There are still semantic and process gaps that need to be covered. And software continue its semantic evolution.

Service types and layers

While the focus is on the business services, there are more service layers. We can easily distinguish between simple and composite service types, but it is even more important to recognize the different service layers.



Note that everything starts from the Business Architecture. Business needs Product Lines. Product Lines consist of Products, which in their turn are collection of Features.

At this point a developer can map Features to Business Services, creating a Business Layer of services.

The hierarchy of service layers is very visible.

Business Layer, such as Order or Customer services;

Utilities, such as Single Sign-On, Search, or Scheduling services, and

Data Layer services that can be called up from Business or Utilities services (but not directly from applications!).

Business services, such as the Order service, are usually named after the business functions they represent. The Order service is usually implemented as a service orchestration or a sequence of composite services responsible for specific processes.

Process services, such as Single Sign-On, Search, Scheduling, and more in their own turn consist of Data services and Utility services, which are often called System services as they specialized in accessing specific systems and data sources.

The art of designing service layers for an application and across enterprise is called today Microservices.

Microservices and API-led connectivity by MuleSoft

Imagine that as a developer you have access to multiple services developed independently and your intention is to select those that provide necessary functionality and connect them into a working application. If you think that it is easy, think again. There is a need for well-structured and well-known APIs, the need that was not well addressed so far.

API-led connectivity by MuleSoft is a good step in that direction. MuleSoft promotes RESTful API Modeling Language (RAML) and developed its own set of MetaData and annotations known as DataSense. Under RAML and DataSense umbrella services are not only re-usable, but can be easily discovered alone with their parameters.

RESTful API Modeling Language (RAML)

Did you work for enterprise that developed thousands of services? At some point you might notice that it is easier to create another one than find an existing service that covers the needed function. This very sad discovery is a good indication that **service discovery** needs improvements.

RESTful API Modeling Language (RAML) is designed to provide these improvements. RAML offers developers a formal way of describing RESTful APIs.

What is RAML?

RAML is built on the top of the standards such as YAML and JSON. RAML is a non-proprietary, vendor-neutral open spec. RAML gives developers freedom of providing their own semantics to define specific properties of services in a specific business domain. At the same time RAML includes basic characteristics necessary to invoke a service, such as **basicUri** (usually serves as the endpoint of REST service invocation), describes the **post** and **get** queries, and **queryParameters** that must be provided with the RESTful call.

Example:

```
#%RAML

title: Course Catalog by Internet Technology School
```

```
baseUri: http://itofthefuture.com/BASE
/catalog
  is: [paged]
  get:
    queryParameters:
      courseType:
        description: type of a course, such as Java, Big Data, Semantic Technologies, and more
responses:
        200:
          body:
            application/json:
              schema: | { "$schema": "http://json-schema.org/schema",
                  "type": "object",
                  "description": "A course type description",
                  "properties": {
                    "courseTitle":  { "type": "string" },
                    "courseInstructor": { "type": "string" }
                  },
                  "required": [ "courseTitle", "courseInstructor" ]
                }
            application/xml:
            application/pdf:
```

This example provides human readable descriptions as well as formal method definitions.

To get a well-formatted response use the PDF type with a simple request:
http://ITofTheFuture.com/BASE/catalog/pdf

Note, that it is up to a developer to choose specific semantics for data property names, such as courseType, courseTitle, and courseInstructor. These naming conventions that look obvious and even trivial for one group of developers might miss expectations of another company, which has a different business dialect. And we will talk about semantic integration a bit later.

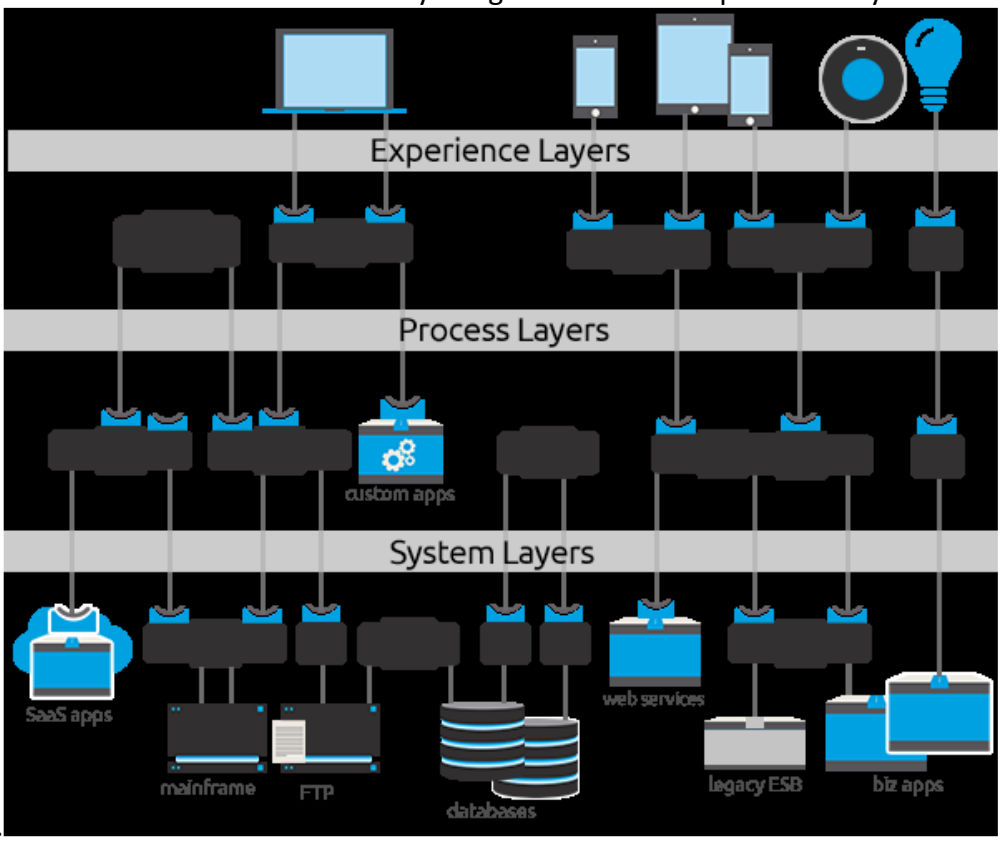
Mule Soft introduces DataSense

Mule Soft made another step in this direction by creating Data Sense metadata for application designers.

MuleSoft is actively moving to Microservices. For developers this move to Microservices means API-led development. This is exactly what MuleSoft offers.

Similar to the discussion on service layers we had before, MuleSoft also separates service layers into three categories: Experience, Process, and System Layers.

The lowest service layer called **System Layer** represents underlying utilities and data services, including APIs to applications that provide data. The Process Layer is responsible for business processes that form workflows and eventually integrated into the Experience Layer consumed by end users.



Data **Sense is a better tool for developers who usually described their design ideas in Power Point and diagrams.** Data Sense allows creating metadata to facilitate application design. Anypoint Studio can understand these metadata and can provide necessary translation data type and structure described there into application body.

At this point Anypoint Studio does some work on behalf of developers. The tool intelligently discovers information about internal and external resources. Usually this was manually done by people. Imagine that there is a mobile application connected to Facebook. Facebook has its own API, data types and structure, which can be captured by DataSense. Anypoint Studio can provide this information back to you, helping you to make better and quicker decisions about interfacing with Facebook.

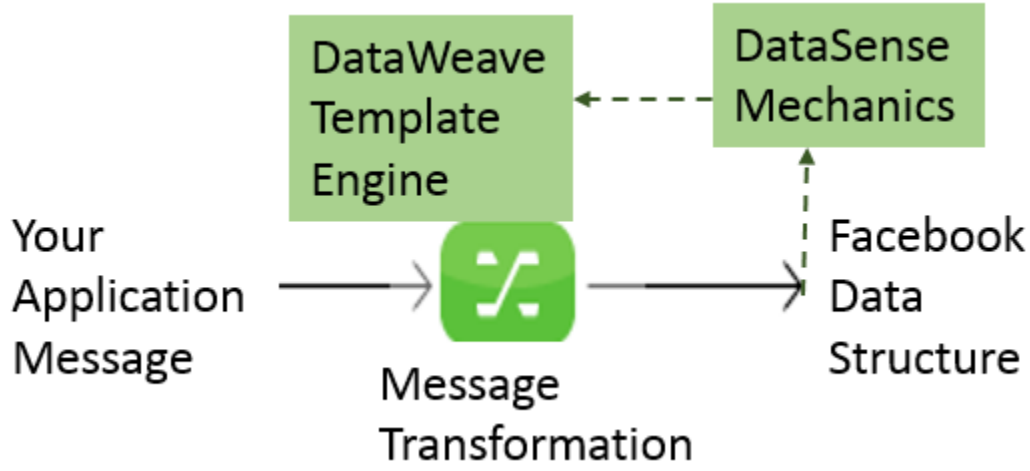
What can be done with DataSense and Anypoint Studio?

In the terms of Anypoint Studio, two major functions to discover and describe metadata are Perceptive Flow Design and DataSense Explorer.

Perceptive Flow Design

Mule can use an existing connection to the resource to retrieve metadata about message properties and payload.

This information will feed into DataWeave, a message transformation component. Then, the mapping data from one format to another happens almost automatically. At least part of the work is done by a computer!

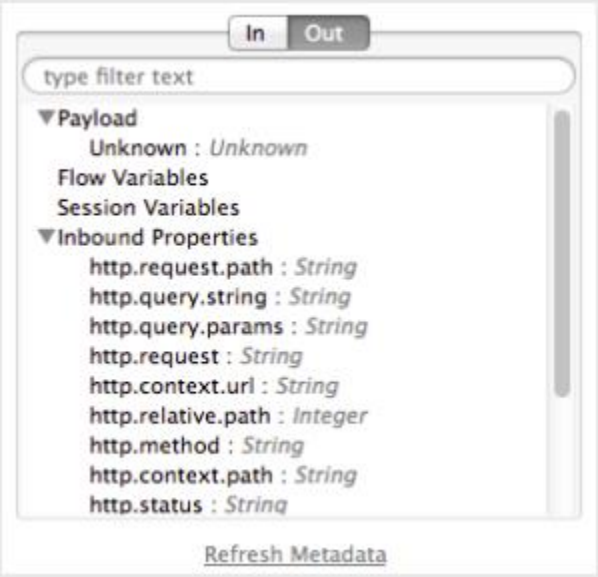


You can type the word ***payload*** in the Anypoint Studio GUI to get a list of all the properties and methods associated to the payload. This is close to magic!

DataSense Explorer

DataSense Explorer is part of Anypoint Studio. Explorer can visualize the message data structure at different points of the flow, when a developer is still designing the flow. A developer can select any element in the flow and the DataSense Explorer will display the structure of input and output data.

With the DataSense Explorer a developer can see the message contents at any given point in the flow. This is possible because the Explorer has access to the DataSense metadata of compatible connectors and knows about Session Variables, Inbound and Outbound and Payload properties.



DataSense and Studio connection

DataSense allows developers to describe and discover information via the connector and connection to the application. Then DataSense passes this information about application entities and their structures to Anypoint Studio. Anypoint Studio presents the data at design time. Studio can even make suggestions about the expected values in fields returned by the connector. These suggestions are based on connector's metadata and DataWeave's intelligence.

What does it mean to implement DataSense?

The implementation consists of the following: **configuring metadata retrieval** by creating the connector to supply this information, and **configuring metadata awareness** with annotations of operations (methods), providing to Anypoint Studio necessary information about the DataSense implementation.

Anypoint Studio is a rich extension of Eclipse with mostly well-known and almost intuitive windows. You can open the Import wizard from the File menu. With a pop-up wizard you can select an existing Anypoint Studio Project from External Location or open a new one. Once you select the Server Runtime as Mule Server 3.6.0 CE or EE, the studio will display the Mule Flows.

For precise settings you can use the ***mule-app.properties*** file with access credentials and more data describing the connector.

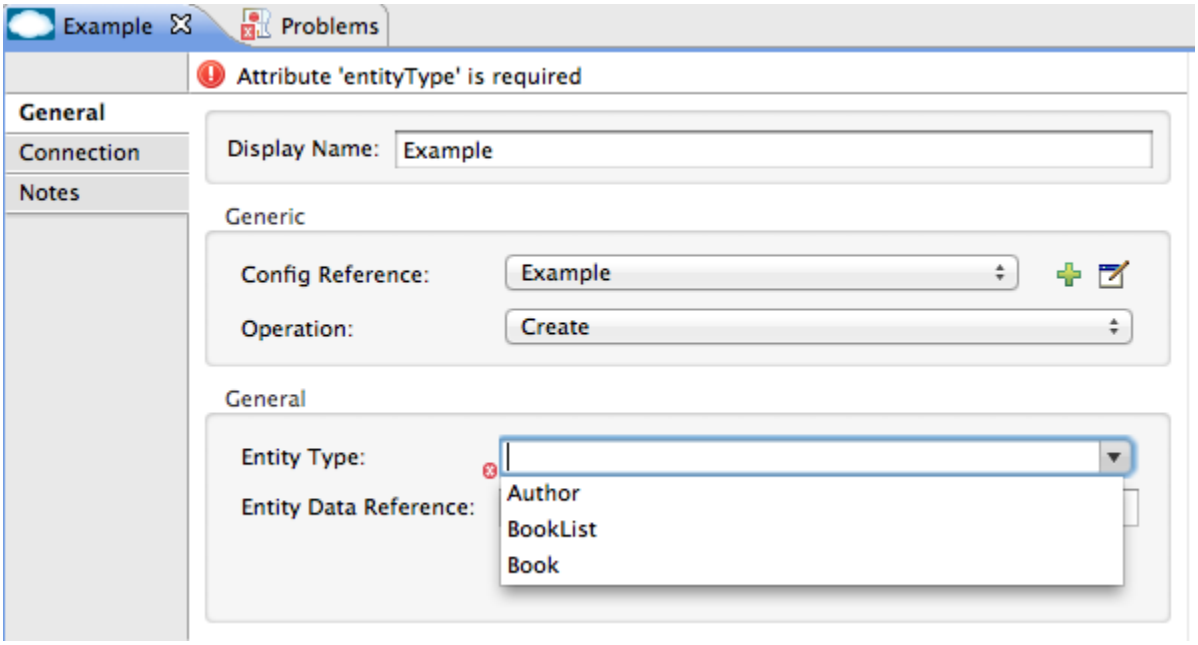
Connectors with Static and Dynamic Data Models

A connector might have a Static "strongly typed" data model or a Dynamic Data Model where data types are resolved at run-time

In the case of Static model, metadata retrieval as well as metadata awareness is immediately available by the strongly typed parameters.

In the case of Dynamic Data Model, some metadata will be resolved at run-time with two annotated methods, getMetadataKeys() followed by getMetadata().

The image below from MuleSoft examples shows one of the GUI windows offered by Studio for configuration.



The bottom line: DataSense and Studio are working together to discover and describe application interfaces. This work saves developer's time and improves precision of the design by bringing an important layer of metadata information at design time.

The trend

Do you see the trend? Development and even its modeling part become increasingly formal. Abstract ideas find their precise expressions and become tangible. Computer programs can understand and test these ideas. In a similar way the abstract concept of interfaces became part of Java language to allow compiler checking for correct implementations.

We describe software program behavior and support this ideology with Behavior-Driven Development tools, such as Apache Cucumber.

And we try to formalize design with simple language, so business people can understand and participate in the design. We did not want to introduce another weird xml languages (like RDF family). This intention is clear. Streamline the development flow and give the business more opportunities to directly participate. This is coming, but coming extremely slowly. Here is just one of the reasons.

New tweaks of technology often require restoring old development arts in completely new environments. For example, application frameworks such as Hibernate and others hide data complexity from us and reduce the need for data modeling skills.

Big Data renews this demand. When millions and billions of records are at stake, perfect data modeling is required. Perfect modeling with perfect understanding of NoSQL DB features and correct expectations of application queries is becoming the must!

There are more reasons telling us that it would be a stretch to expect business masses joining us in development trenches next week or next year.

So, what is the next step of software evolution?

The next step in the software semantic evolution

In the beginning was the Word…

One of the earliest known civilizations was Sumer, in the Uru region of the Middle East (now Southern Iraq), about five thousand years ago.

The Sumerians soon dissolved into the Chaldeans, Jews and Babylonians, but not before developing a system of numbers and writing, which is the foundation of the system that we use today.

The number of Sumerian glyphs was between 400 and 1000. They represented words or small parts of words.

Chinese language has from 40,000 to 80,000 characters (hieroglyphs), depending on which dictionary you pick.

The Kangxi Dictionary describes about 40,000 characters, while the modern Zhonghua Zihai shows nearly 80,000. This number is comparable to the number of words in a modest English dictionary.

Chinese characters represent words and complex concepts, sometimes phrases and even sentences, an “all-in-one” communication solution.

A significant development in human history, languages went a long way towards optimizing the expression and communication of thoughts, ideas and dreams. Alphabets represent the smallest pieces that can be used to form words, and further combined to construct sentences, articles, journals and books.

Can you see a clear analogy in the software world? From “all-in-one” programs to smaller and smaller pieces.

Looking for ways to communicate with computers, computer languages started with numbers and evolved to using English words to describe variables, properties and operations.

(Find more about the history of computing and computers here: http://itofthefuture.com/BASE/Lookup?action=content&issueID=5.)

But the ultimate communication tool is still our living language. There is no more powerful alternative to that flow, created over thousands of years.

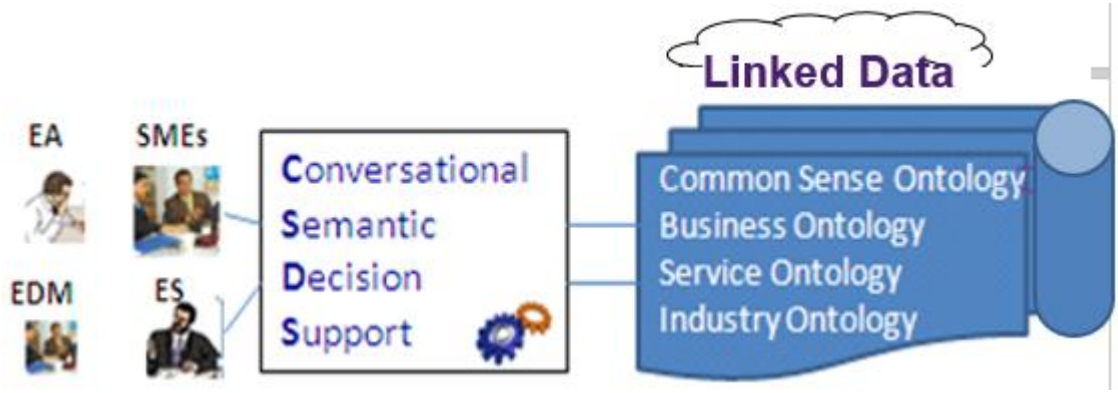
We are coming closer to the point when computers will understand natural language.

Then we will be able to establish a new type of development.

The development, modeling and manufacturing processes will be available to a non-technical person who has creative ideas but not the "know how" details.

What we today call design and development will transition into a direct conversation between a person and a sophisticated computer program which can be called "a modeling and manufacturing factory".

Initiated by a person and supported by the conversational semantic system, these conversations will help to clarify the initial ideas. Whenever the system has a hard time understanding a human, the system will start asking a set of clarification questions.



Developers at any enterprise: Enterprise Architects, Enterprise Data Masters, Enterprise Service developers and all kind of Subject Matter Experts will initiate a conversation and with system support will be able to develop a working application.

The system of **Conversational Semantic Decision Support** is based on several ontologies and conversational scripts, prepared up front to clarify human expressions that cannot be immediately resolved to concrete understandings.

The word **Ontology** has several meanings. Here it is a computer file with representations of knowledge, focused on a specific domain and organized in a graph of Linked Data.

The biggest challenge is obtaining **Common Sense Ontology**, which helps in understanding human expressions. The most expressive Common Sense Ontology has been developed by Cycorp [1], but their heavy-weight knowledgebase and the mechanisms of handling ontology are far from the current RDF-based mainstream of semantic technology. (They started long before the mainstream became mainstream.)

Business Ontology reflects corporate business specifics, corporate rules, policies and processes. Some companies, such as Sallie Mae and Wells Fargo, have developed or are in the process of developing their Business Ontology on top of **Industry Ontology**.

The financial industry was the first to create a standard ontology that reflects financial operations.

Due to the government support and direct order, and to collaborative efforts by the Enterprise Data Management Council (EDM Council) and Object Management Group (OMG), the Financial Industry Business Ontology (FIBO) [2] has been released as a series of standards, providing a description of the structure and contractual obligations of financial instruments, legal entities, market data and financial processes. (I am proud to be a participant in this work as a member of the FIBO technical committee.)

Service Ontology is a semantic graph of services with their descriptions. Think of a semantic service map, which can be developed for conversational interaction [3].

The person will be working in collaboration with the system to model and manufacture the desired implementation. The computer system will have access to “know how” details and will be able to add more to its library (think of services and Microservices) as a result of conversational interaction with the computer system.

This Knowledge-Driven Architecture [4] is the optimal combination of humans’ ability to suggest new approaches with computerized translation of these ideas into properly formatted, executable instructions.

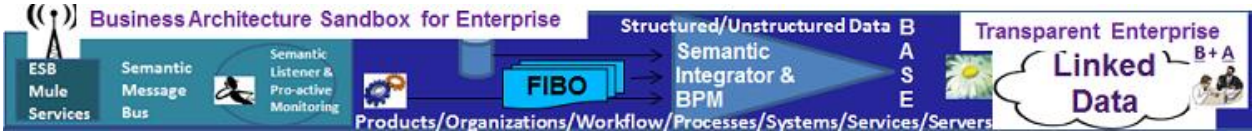
The conversational system will search all available knowledge domains and in the difficult cases come back to a SME with clarifying questions. Eventually, they (SME and the system) will be able to successfully model and implement the idea into a product and manufacture the product with the tools similar to 3D-printers.

An important feature of the Conversational Semantic Decision Support system is the benefit of naturally growing Ontologies and conversational scripts as the result of conversations. The benefit, which we, people, also have in the most conversations. Yes, it is a two-way street!

Read more about conversational development and the changes in technology and society in the nearest future at http://itofthefuture/book/message.pdf.

This is not just a dream. The book online, “IT of the future”, http://ITofTheFuture.com, [5] focuses on practical steps transitioning from the current complexity of enterprise to Semantic Cloud Architecture. The book describes the way of carefully placing the seeds of the new technology in the current business ground and potentially getting about 50% of budget saving in the process.

An important instrument for such process is a common playground for developers and subject matter experts. A working prototype, Business Architecture Sandbox for Enterprise (BASE).



A web application, integrated with Mule ESB, BASE includes one of the lightest versions of FIBO as Industry Ontology and provides the way of creating Business Ontology and Service Ontology on the fly, while creating services and workflows. The main focus is on engaging subject matter experts in the new paradigm of creating business processes and workflows on-the-fly with some limited collaboration with developers.

BASE is not positioned as a product, but as ideology to follow. The book describes a great deal of technical details of BASE helping to re-create and customize the tool.

I will come back to BASE a bit later. At this point, I would like to briefly review the steps in the technical ladder leading us to the semantic revolution.

SOA ruined the **all-in-one** programming paradigm and shifted development focus from applications to services.

Microservices fight for independence against application flavors.

Independence is expensive. While getting rid of application specifics, we decrease the essence of a service.

Trying to play well in any environment, the shell of the service, the frame of the service package, is getting thicker.

There is an associated cost and potential profit in years to come. While associated cost is well visible, it is harder to estimate Return on Investment (ROI). Unfortunately (or fortunately) accumulation of changes in technology and business direction throw away whole systems or brings a new development paradigm to redo them. This happens every three-five years.

Massive Cobol systems are still running on mainframe not because of that technology superiority. They are just "too big to fail". Symbolizing more liability than profitability, these heavy-weight old systems are hard to replace in one shot. This requires long term mentality and provisioning, which is also hard to find today in the corporate world.

The bottom line: Microservices is the right step in many cases, although not in all. One of the benefits, which is a very important one, they make our constructions less monolithic, lighter and easier to change.

Smaller pieces and bigger variety of them require better handling tools and more automation. This is still coming.

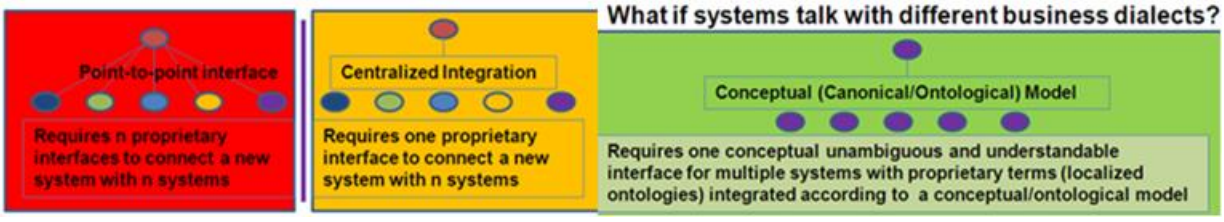
RAML introduces a semantic flow of technical descriptions of API, which improves the way of handling services.

DataSense by MuleSoft adds important metadata language that adds to this semantic flow of software design. Each company chooses their own naming conventions.

While these naming conventions look good for one business, they might have different names in another business. The next step is to prepare these services working across several businesses with different business dialects. This can be done via a canonical semantic data schema, or more precisely via the semantic graph, a semantic integration layer.

A Semantic graph can represent a business domain, providing canonical object names with their synonyms and connections between objects and their properties. The semantic integration layer serves as a formal data dictionary for choosing the names, which will work across multiple business dialects in the same business domain.

The illustration below tells the story of the integration evolution, from point-to-point to centralized integration with Enterprise Service Bus (ESB), and further to canonical interfaces with the semantic layer, which connects multiple business dialects.



This semantic layer will provide mapping of proprietary data to the Canonical Data Model (Common Ontology) language. This is an important component of system integration. This is also essential for designing API for 3-rd party developers.

Enterprise Service Bus handles the messages from many services and applications. To subscribe for a message or a topic any subscriber needs a precise description of a specific message or a topic. Such descriptions are usually very technical by their nature.

The semantic layer on the top of ESB will change the way of handling enterprise messages.

This layer will allow developers to introduce a **semantic listener program** and provide opportunities for subject matter experts to talk business terms while expressing their interest in specific reports based on enterprise messages.

And this is another step in the right direction: preparing a semantically-rich enterprise environment.

By providing meaningful service names, descriptions, and messages, developers establish better connections between business functions and their technical implementations. Semantically rich environment improves search for people and computer programs in multiple areas: root-cause analysis, business process modeling, creating and managing applications. This is the direct connection between business requirements and API-led development.

One example of a direct interaction between business, developers and ontology can be seen with the setFinalPayment() operation/method that is defined in the FinalPayment service. The FinalPayment is one of the existing concepts in the Financial Industry Business Ontology (FIBO.) [2].

By sticking to the names describing business processes in FIBO, developers, architects and business analysts, working in financial industry, will come closer to a common language that is the key in improving business efficiency.

Semantic Logging and Semantic Listener

In a semantically rich environment, there is no need for complex monitoring tools. The service names and descriptions as well as application messages are self-explanatory and directly tied to the semantic execution model.

Application messages can describe as many properties as necessary with the idea that each property is defined in the semantic model. The messages can tell the story about WHEN (time), WHAT (description of the event), WHERE (system or/and service name), HOW Serious (type), HOW to fix (recovery action), and WHO should be notified.

A relatively simple **semantic listener program** can understand and **act** upon these messages.

This approach, when it is consistently used across the company and industry, will create smaller, smarter, and inexpensive semantic-sensitive tools to monitor and manage service operations. The same message will become a valuable record in the root cause analysis and recovery processes. Such records can be RDF-formatted. These RDF-formatted records-messages can represent the “situational awareness” factors.

Business Architecture Sandbox for Enterprise

The next step of software evolution offers new opportunities in many areas. One thing is clear: with the volume of information doubling every year, and with increasingly interconnected departments and corporations, semantic technology, the cool new kid on the block (who also happens to be pretty darn smart) is well on its way in.

In the future, new class called Knowledge Engineering and Semantic Cloud Architecture will be introduced in every school along with the subject of Critical Thinking. Modeling tools that have Business and Development views today will add an Ontology view tab to the front page. This is happening as you read these lines.

Semantic technology helps computers to better understand unstructured text, not just our commands. Then computer programs greatly increase their ability to partner with people on decision-making processes.

But stop dreaming of Artificial Intelligence. We are not there yet. Computers can help us more … when we can help computers. This is about a conversational approach, when a program is not necessary smart enough for complete understanding, but as a child can ask a clarifying question.

This is about a new generation of systems built with **knowledge-driven architecture**. [4].

A good example would be adaptive robotic systems that can learn by conversing with people and store new skills as orchestrations of services.

A fundamental problem of current robotics is their limited set of skills that hard to expand. This is related to the current development methods that require multiple translations from natural language of task requirements to compiled and integrated working systems. Current robots are programmed to perform relatively simple, well defined and predictable tasks.

Adaptive robot system [6] with knowledge-driven architecture includes a built-in conversational mechanism to translate on-the fly changeable situational requirements into close to natural language but more precise terms. Each successful translation introduces another rule or even a situational scenario, adds a service, and increases the system power.

The integration of software and knowledge engineering is arriving on the scene in much the same way that object-oriented programming did when it replaced structural programming.

Similar to that time, the gap between the realities of the current enterprise and Semantic Cloud Architecture seems so huge that most companies are very cautious in approaching this cliff.

Business Architecture Sandbox for Enterprise (BASE) was designed to minimize this pain and to plant the seeds of Big Data and Semantic technology in the current business ground, enabling the next technology revolution.

BASE runs as a Web Application integrated with Mule, ESB [7] and Apache ActiveMQ [8]. This integrated system is configured as a cluster with multiple servers, providing high availability and failover.

BASE allows developers and subject matter experts describe and create business processes and workflows based on the REST API created on-the-fly on the top of business ontology.

These basic SOA standardizations provide the ground for service orchestration, reducing tight coupling of applications, and decreasing production problems and maintenance efforts.

To play with the prototype online just use the descriptions, the link and the key in the full version of book online.

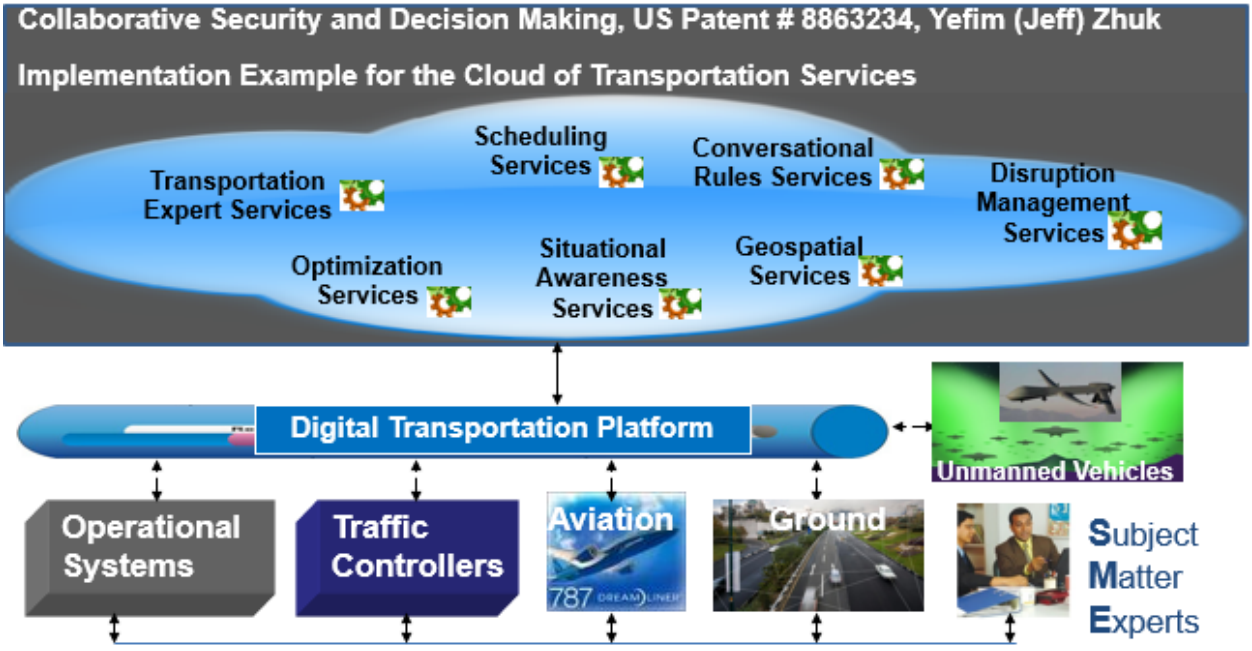
Collaboration of Services and Transformation of "tribal knowledge"

Collaboration between people and groups seems to be a thing with a positive sign, although we know how difficult this can be. Distributed knowledge and process systems [9] allows involved parties, people and companies, negotiate multiple forms of collaboration online while sharing data and services.

What is the need for collaboration for services?

Collaborative security of service groups is different from a single service security.

Simultaneous activity of many services, working on a common task, requires collaborative decision making. Think of a situation with multiple transportation services on the ground and in the air, when their interaction and collaboration is the must.



How can computer services optimize their behavior, when many of them simultaneosly perform different and sometimes conflicting tasks, interfere with external events and weather, trying to adapt to a quickly changing situation?

Collaborative Security and Decision Making in SOA environment [10], answers this question and turns this beautiful idea into a working system.

One of the keys, is a multi-dimensional system of rules driving service behaviour. Another key, similar to people's collaboration, is the ability of system services to converse, understand, and

adapt to the changes by adding or updating the rules. The difficult part is the mixture of business and technical slangs in expressing events and situations.

Generally speaking, business prefers natural language, while technical language is XML and web services standards. Necessity of the semantic bridge is obvious. The bridge is coming especially handy when Subject Matter Experts must intervene in an unexpected situational scenario.

What is the source of rules and how to establish correct rules for a selected rules engine?

Current practice answer this question by calling consultants. This is not only expensive. The biggest problem is that consultants do not know specifics of the business, the knowledge domain that is essential for creating the rules.

Some knowledge can be retrieved via published resources, corporate regulations and policies. But the research shows that about 70% of information is so called “tribal knowledge”, never computerized experience of subject matter experts.

The **Rules Collector system** [11] helps capturing the expertise of an individual in a formalized manner as a set of rules for a selected rules engine. The transformation happens over a long process initiated by a program to retrieve a complete information from a subject matter expert, sufficient enough to be formalized as a rule.

Yes, a computer conducts an interrogation of a Subject Matter Expert (SME), clarifying ambiguous expressions and connecting the dots, word by word.

At this time of massive retirement of the “baby boomers” in various industries, capturing their “tribal” knowledge becomes one of our most important tasks.

Capturing corporate knowledge in a computerized form is a pre-requisite for the next step in the development process, when the “know how” will belong to the computers.

Less technical translations and translators will be needed, and many more developers will come up with creative ideas for this exciting development stage.

In the beginning was the Word…

References:

1. Cycorp combines an unparalleled common sense ontology and knowledge base with a powerful reasoning engine and natural language interfaces, http://cyc.com
2. Financial Industry Business Ontology (FIBO) standard, http://www.edmcouncil.org/financialbusiness
3. Conversational Semantic Service Map, Yefim (Jeff) Zhuk, The system for collaborative design, assembly on-the-fly, execution, benchmarking, and negotiation of computer services and applications by developers and subject matter experts, US Patent Pending.
4. Knowledge-Driven Architecture, Yefim Zhuk, Streamlining development and driving applications with business rules & scenarios, US Patent, http://www.google.com/patents/US7774751
5. The book online, "IT of the future", http://ITofTheFuture.com, focuses on practical steps to transition the current IT of competing applications to a unified Semantic Cloud Architecture and describes Business Architecture Sandbox for Enterprise.
6. Adaptive Robot System with Knowledge-Driven Architecture, Yefim Zhuk, On-the-fly translations of situational requirements into adaptive robot skills, US Patent, http://www.google.com/patents/US7966093
7. MuleSoft Enterprise Service Buse (ESB), https://www.mulesoft.com/
8. Apache ActiveMQ, http://activemq.apache.org/
9. Distributed Knowledge and Process system, Yefim Zhuk, The system allows negotiate multiple forms of collaboration, and contains sufficiently flexible levels of data security for online collaboration, US Patent, http://www.google.com.sv/patents/US7032006
10. Collaborative Security and Decision Making, Yefim Zhuk, transforming a beautiful idea of collaborative security decision making into a working system, US Patent, http://serviceconnect.org/
11. Rules Collector system, Yefim Zhuk, Transforming "tribal knowledge" into formal rules to drive applications and business processes, US Patent, http://captureknowledge.org/

Read more in the book…

Part 4: Big Data and Semantic Tools at Work

The most important task on the list
Review the tools for the task
Cognitive Computer Foundations
Knowledge-Driven Architecture with Corporate "Know-How"

While Big Data is a relatively new concept, the exponential growth of information is a very old, well known process. This process was drastically accelerated with the addition of another information channel, the Internet. Naturally, Google became one of the first among the ideologists and practitioners dealing with this phenomenon. Many followers expanded the original ideas of Big Table and Map – Reduce and brought new ideas to the mix, which we currently call Big Data industry.

Big Data allows us accelerate information processing while creating more flexible data structures. But structured data is only a small fraction of information. What can help us understand semantics and process unstructured data?

Let us review the intersection of Big Data and Semantic tools, the informational space and direction that can help us in what I consider the most important task on the list.

The most important task on the list of Information Management

More than 60% of the working population is eligible for retirement and the number is growing. Replacing "experienced and expensive" with "young and cheap" is a common business process. "Nothing personal – it's just business."

So, what is the business side of the story?

More often than not, a company gets short time advantage from the direct financial cuts. Its stock usually goes up for a while. But the future of such a company is not clear. Its "tribal knowledge" has been lost. The pain is real, especially for the companies dealing with long-life products, which are surrounded by a monstrous flow of related rules and regulations.

Read more in the book about Big Data and Semantic Technology tools working together …

Review includes the descriptions and comparison of the following technologies that are currently used in many analytic tools, such as Jasper Server and more:

BigTable, Hadoop and Map-Reduce, OWLIM, AllegroGraph, Neo4j, Fluid Operations (fluidOps), Cassandra, MongoDB, RavenDB, Kafka and Storm

Based on events of 2040: The response that comes afterwards

The report predicted that the modeling factory production will continue slowing down until they reach some critical point that we passed several months ago. This will result in a violation of the agreement between the company and the clients. This might be the end of the company...

"Any constructive idea? Anyone?" - This was the president. - Silence was the answer.
She looked at me. There was the case in the past, when I suggested something that actually worked. Usually I was just good at asking questions and generating discussions. The president preferred keeping me close during the meetings like that. Although, no meetings like that have ever happened.

I did not have any constructive idea and started with the questions to the psychologist.
-"Should we trust the robot's conclusion? Can we have a second opinion on technical and psychological aspects?"

“Taking into account our timeframe, I would say “no” to both your questions” - the psychologist smiled, and her smile was very sad.

“Can we limit robot's collaboration by some self-adjusting rules? Or maybe gothe opposite direction? Can we provide multiple knowledge domains in each robot, so less communications would be needed?”

-”We already tried new rules. It did not help, just created more traffic to measure and evaluate effectiveness of communications. Initiating multiple knowledge domains or making “super-robots” is prohibitively expensive”. - That was our technical advisor.

Several people questioned how much we should trust the report. Could it be an intentional plot? What would be a motivation? Who can benefit from this scenario? The discussion made a full cycle and dried out. I did not want to believe in the conspiracy theory with the robots. My preference would be to think of the technical and psychological problems, trying to fight complexity with a simple solution...

-"Miss President, What could be the consequences if events follow the pattern suggested in the report?"

-"Public outcry will be immediately supported with new regulations "to protect consumer rights and regulate the modeling factories."
-"We will be obliged to uniformly follow the regulations regardless of circumstances."
-"All the changes we currently make on-the-fly would be approved by regulatory organizations."
-“This will significantly slow down or even kill the company.”

The picture was terribly clear and real.
-"We might have a chance for a preventive action" - the psychologist seemed to recollect something important.
-"Sometime ago I had a conversation with Provident…"
-"Provident suggested an interesting plan of actions. I can describe the main idea, but we might greatly benefit from his participation."
I was always intrigued by their relationships, but Monica was the only person who had at least slightest understanding of Provident plans, ideas, and even whereabouts.

The idea was amazingly simple.
According to Provident’s theory, corporations accepted regulations because they did not do any better without.
Regulations are usually a response to business pain points. The response that comes afterwards, when damage already done, restricting the business and sometimes even killing it, while trying to prevent the situation from the past.
Provident suggested a set of actions that could be more efficient, pro-business oriented, while looking more into the future, then into the past.

Analyzing past crisis and addressing new pain points should become a business goal for a company or an industry

A corporate business will announce a start-up competition on achieving the goal and allow initiating start-ups by any group or a person within or outside of the corporation
Business selects several groups and supports them by matching some percentage of the resources provided by the startups
Government will also support the startups by providing for each group a "super-robot" trained in multiple knowledge domains. While humans of different team rarely communicate their ideas, robots freely exchange information and help each other.
Provident expected tremendous return on investment, but as far as I know this was never done.

"This sounds interesting, but..."

The president interrupted me: "This sounds like a chance. Monica will connect you. Apologize before him and ask for help. He might like the opportunity to implement his ideas. Go, time is ticking."
Monica slightly nodded. We quickly left the meeting.

Read more in the book…

Part 1: Knowledge Driven Architecture | **Part 2: Transitioning to Semantic Cloud**
Part 3: After SOA: "What, Why and How" – Software Evolution – The Next Step
Part 4: Big Data and Semantic Tools at work
Part 5: Semantic Toolbox and its Magic for Validation of Development

---

The message from 2040 | **Discussions with the first readers** | **Buy the book**