The next step in the software semantic evolution … Conversational AI

*With thousand thanks to Keith Miller, InfoFusion, the first reader and editor.

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.

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

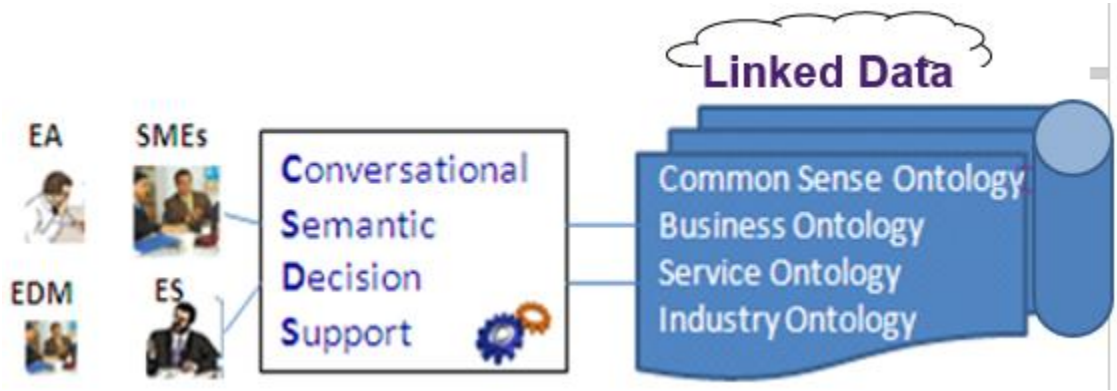
We are coming closer to the point when computers will understand natural language.

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 that I participated 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.

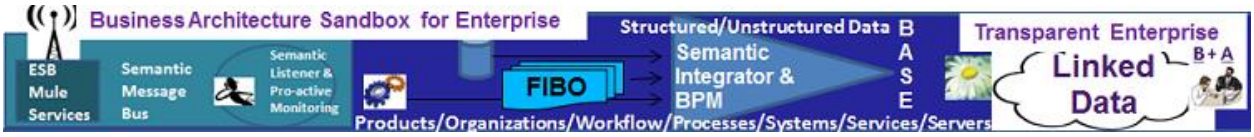
The conversational system will search all available knowledge domains and in the difficult cases come back to a SME with clarifying questions. Eventually, they (SME and the system) will be able to successfully model and implement the idea into a product and manufacture the product with the tools similar 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. Getting rid of application specifics, the essence of a service become smaller.

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.

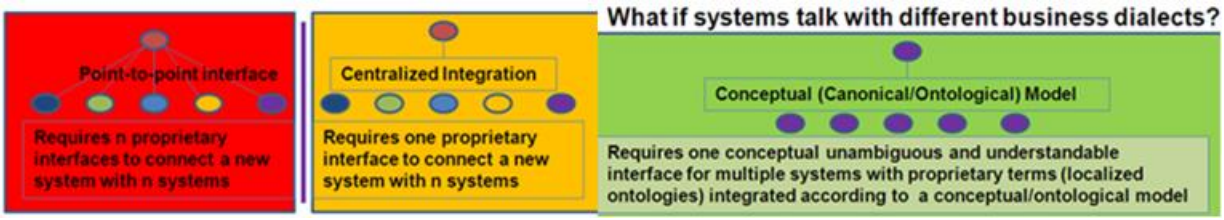
RAML introduces a semantic flow of technical descriptions of API, which improves the way of handling Microservices.

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: 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**. [2].

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.

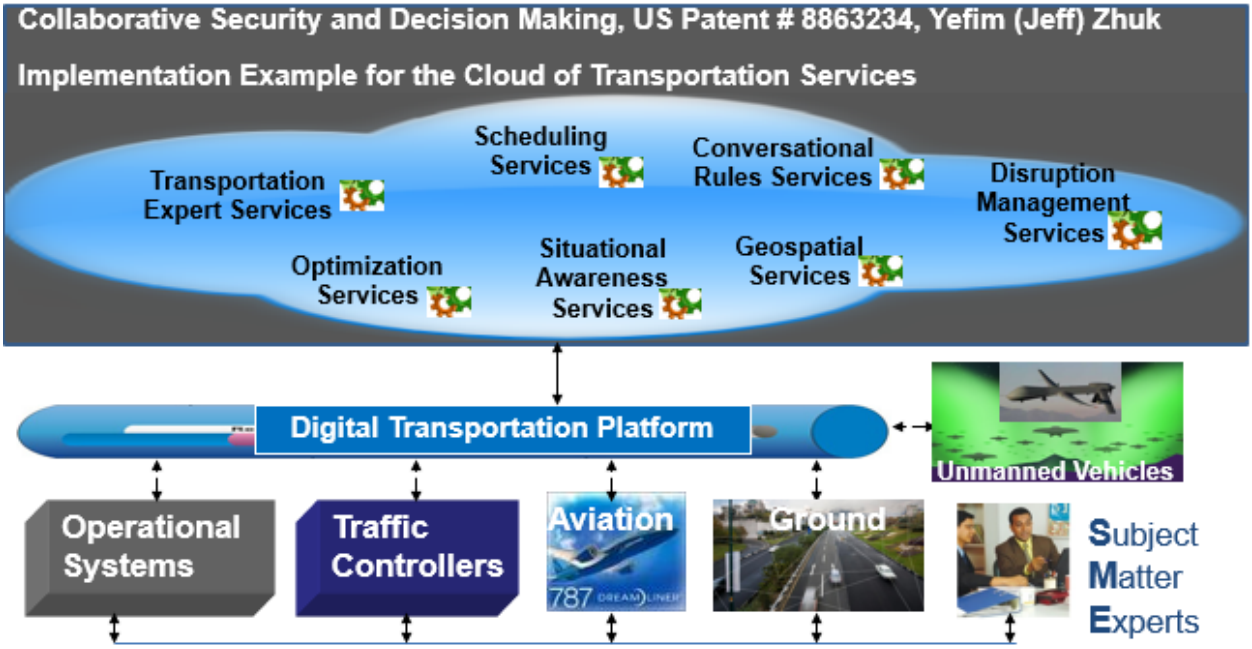
Collaboration of Services and Transformation of "tribal knowledge"

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. Development Factory, 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, 10956676B2, https://patents.google.com/patent/US10956676B2/en
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, acquired by Yahoo, Inc. 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, patented in the US and 15 European countries, acquired by Boeing Co., http://serviceconnect.org/
11. Rules Collector system, Yefim Zhuk. Transforming "tribal knowledge" into formal rules to drive applications and business processes, US Patent, acquired by Boeing Co. http://captureknowledge.org/