Top Development Skills

About the author: Jeff (Yefim) Zhuk worked for Boeing and Sallie Mae, consulted government agencies and corporations in SOA and knowledge engineering, shared his expertise at Java One, Semantic and Boeing conferences. His publications and patents describe a field of Integrated Software and Knowledge Engineering.

An experienced trainer, he has created a carefully selected set of courses helping beginners and professionals to climb a ladder from fundamentals of software development to the highest-in-demand skills and exciting jobs as a AI developer, consultant or entrepreneur.

Hobby: Mountaineering - http://Everest6500.com

What development skills are in demand today and will be in a bigger demand tomorrow?

The answers can be found in the cross-section of several related avenues.

1) What are the most important business goals?
   - Productivity and profit.

2) What products and services have the biggest distribution and the best ratio between the cost of production and market price?
    - Software and knowledge – although expensive to produce – can be distributed as a product or a service to millions and even billions potential users

The key to increasing productivity is automation with AI components. This does not necessary mean that only software developers who create AI tools are in the driving seat. Any person who understands perfectly well her or his daily routine can translate this knowledge into automation … after learning the art of translation.

Another important factor – mass-distribution – depends on the art of marketing. Marketing and sales services are extremely valuable … and can also be enhanced by using AI components to find better intersections between the market and the audience. AI components help achieving this goal by taking into account more details and composing a graph of inter-connections.

AI increases productivity in every area including marketing and sales. AI opens new horizons that we did not see before. We recognize new opportunities and start doing new things, things that we could not even imagine today. This spiral leads to even more opportunities – translated into more jobs, which require new skills. Increasing productivity means increasing paycheck and improving quality of life, especially for those who acquired the skills.

The bottleneck is our ability to quickly learn and change. The current formula of education is not sufficient. Colleges and universities are far behind the technology curve. Translation of expertise into educational materials with the following accreditation take years and years.

Do we know any better? Actually, yes! Internet Technology University is developing an AI Training Platform to allow more people to quickly adapt to changing markets and move from low to highly-paid jobs.

These skills are going to be a common ground for many professions.

Even in the software development we came close to another turn where a new set of skills is becoming priceless. While almost every profession will be

enhanced with a fraction of AI, the most complete spectrum might be found in software, which produces AI instruments and services.

So far, software serves as a platform for AI. Let us take a closer look at its transformation over a brief history.

A brief History of Software

With SOA, Microservices, Semantic Evolution and Artificial Intelligence Components

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 systems that we use today.

The First Software *all-in-one* Programs. Many years after the Sumerians, the first computer was released: the Electronic Numerical Integrator and Computer (ENIAC) in 1946. Software programs for the early computers included everything: the hardware drivers, the data for the software, the business logic required to run the software.

All these pieces were tightly integrated and mixed together in order for the program to work. In the beginning of my own career, I wrote programs like these, first in binary code, then in Assembly, and later, as software progressed, in programming languages such as FORTRAN, COBOL, C, and FORTH.

Object-Oriented Paradigm (OOP) and Layered Architecture. It took the industry several decades to transition to the Object-Oriented Paradigm (OOP) and Layered Architecture. While the early software programmers had to write hardware drivers, programs that communicate with the computer hardware, telling the hardware how to work properly, the later programmers

did not have to worry about that part of the program. Operating System and Database vendors took over the hardware driver and database programming, leaving the software developers to focus on the application layer, creating better-working, more complicated programs.

Service Oriented Architecture (SOA). SOA shifted development focus to business functions and related services, with the idea that applications must be designed as reusable connected services. This idea taught programmers to pay more attention to business specifics and build versatile pieces of software that could be used for many applications.

Microservices. Microservices further helped to more precisely define services while fighting for service independence against application flavors. Independence is expensive. Getting rid of application specifics, developers decreased ?the essence of a service while increasing the frame of the service packages.

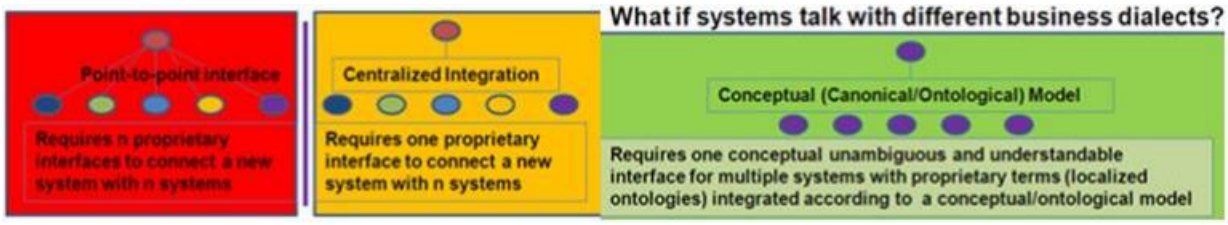
API-based services and RAML. Application Programming Interface (API) became a standard way to introduce services.

RESTful API Modeling Language (RAML) and Data Sense by MuleSoft provide a semantic flow of technical descriptions of API, making it easy to introduce and manage services and microservices.

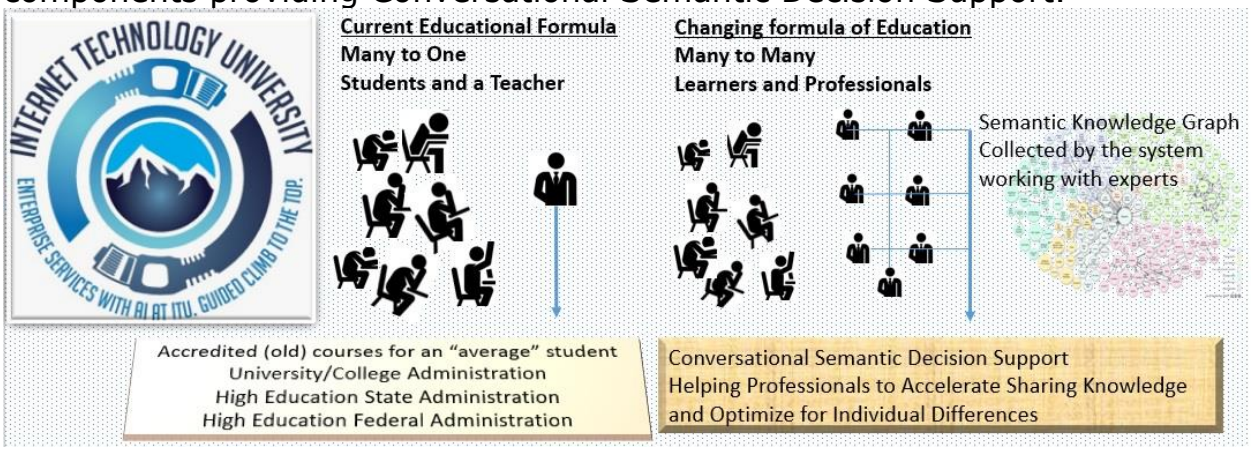
While these naming conventions look good for one business, they might have different names in another business. The next step is to prepare these services working across several businesses with different business dialects. This can be done via a canonical semantic data schema, or more precisely via the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



For educational domain semantic layers can completely [change the formula of education] with individual approach to every student. How? With the AI components providing Conversational Semantic Decision Support.



Enterprise Services with AI components

Artificial intelligence can mean many things. I focus just on one. Computer programs are becoming more helpful. They start working for us not just as stupid machines, but almost as partners.

In partnership, it is extremely important if partner understands your intentions and plans. This is a big "if", which is gradually dissolving into "how".

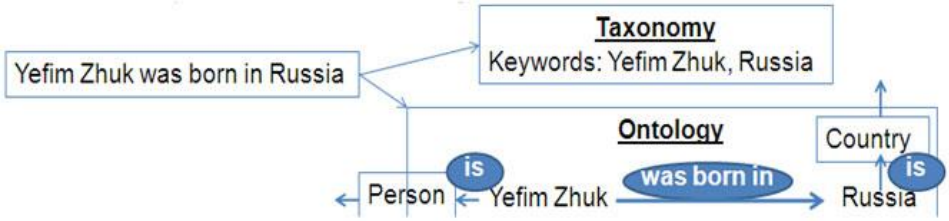
A semantic graph is the mechanism to describe associations or connections between different objects with the idea that this is the richest representation of a knowledge domain, often called Ontology.

This might be a good point to discuss the difference between **Taxonomy and Ontology**.

Just two words about ontology. 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. In a way **Ontology** is similar to **Taxonomy**, but there is an important difference.

Taxonomy collects keywords to describe content. Ontology uses more powerful methods to create more detailed meta-data models. In addition to collecting the key vocabulary, ontology picks up on the relationships between the keywords effectively building a semantically rich and much more meaningful model of the content.

For example, in the sentence "Yefim Zhuk was born in Russia", taxonomy would only use two keywords from the sentence, the name and the place. Ontology would also include the relationship between the two keywords and create a graph representing the content in a greater level of detail. This graph can be extended and later used for querying, in other words for asking questions related to this information, such as "what is Russia". . .



By being more formal and detail oriented, ontology helps us elevate information to the next level understood by computers.

And that fundamentally helps a computer to understand what service we are looking for, what workflow and application we would like to build and etc.

For example, 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.

Knowledge-Driven Architecture*. The challenge, still, is the gap between the business and the programmer: business language is very different from XML, service terms, and programming languages. Knowledge-Driven Architecture is a new way of architecting systems based on business rules and scenarios. This step requires a new type of a developer - one who understands the semantics of business and can clearly express new ideas bridging the gap between the software, and its actual, practical use in the corporate or research worlds.

Today we can see new software frameworks, such as Google Robot Framework or Cucumber. We will briefly discuss them below.

In the beginning was the Word …

Under SOA umbrella, professionals are engaged today in **Microservices**, **REST API Modeling Language (RAML)**, and the Metadata tools, such as **DataSense by MuleSoft**, and others that extend software semantic evolution. Read more about Software Semantic Evolution here: http://www.dataversity.net/software-semantic-evolution-and-the-next-step-part-1/ [1].

I am a hands-on enterprise architect, consulting on integration of software and knowledge engineering [1] and rule-based applications and teaching architecture and development of web and mobile apps.

I was always amazed that we invest so much time into new and better ways of creating dumb old things. Yes, it is still the same inventory program, but it is done nicer and works faster this time! Does it sound familiar? Isn’t this the mainstream of software development?

Slow and cautiously, the mainstream is moving to new territories. Google, Amazon, Apple, IBM, Tesla and some other companies (not too many so far) are working on robots, drones and self-driving cars, competing in the area of **artificial intelligence** (AI).

For a long period, AI lived on the bottom of the lake of opportunities. Recent years turned the lake into the ocean and the underwater current brought AI back to the surface. Nothing else is growing so quickly with the demand for new skills and talents.

Artificial intelligence can mean many things. I focus just on one. Computer programs are becoming more helpful. They start working for us not just as stupid machines, but almost as partners.

In partnership, it is extremely important that partner understands your intentions and plans.

Can we express in computer terms what we want to do?

This is a big "if", which is gradually dissolving into "how".

There are new approaches that appear almost daily in the growing world of Semantic Technologies.

But, as we will see a bit later, tools are not as helpful without the fundamental knowledge, which we luck today.

Information Technology is looking for Big Data, Semantic Technology and Cognitive Computing skills, but colleges and universities still offer Visual Basic and C++ programming.

No questions, developers still need to know programming languages. We have hundreds of them today.

Which one to learn? Almost all Big Data and Artificial Intelligent products, including the famous IBM Watson, are created in Java, which can easily run on any platform, from mainframe to mobile phones.

While teaching at University of Phoenix, I suggested to change the curriculum, but any change take years for accredited schools. Only price change can be done quickly there.

Fortunately, there are valid alternatives to enormously expensive education in colleges and universities.

Online study is helping many people across the world in the learning process. There are several websites such as Lynda.com, Udemy and Total Training, which provide access to thousands of course libraries.

However, most of them have the following common limitations of online learning.

- One approach for all students

All students are different, which is why one single tutorial on a subject might not be able to answer all the questions that each individual student might have. This can become quite a problem for those who need to ask a question, but do not have the right resources and left to their devices to find answers.

- Too Many Useless Connections

When investing time to study from a tutorial, students realize that it had little to no relevance to the subject matter – or added very little value to their knowledge.

- No Hands-On Experience

Online tutorials in most cases hardly give students any exposure to real-life projects, just providing simple, often not even complete, examples.

To combat these issues, many students are now opting for different resources such as Internet Technology University, http://ITUniversity.us, (ITU) where they can get their personalized guidance from an experienced instructor and work in small teams on real projects.

ITU carefully navigates around thousands of existing subjects and tools and provides a selected set of courses that help building the ladder to the latest internet technologies. At each step of the ladder, students build their confidence by working in small teams on real projects. This set of courses is well integrated and optimized for the area of the biggest demand of the industry giving the students great chance for a job and a career.

ITU uses innovative methods of education described here: http://FixingEducation.us [2].

These methods focus on soft skills, such as art of **Critical Thinking** and **Communications**, important pre-requisites to Knowledge Engineering and to the science of Integrated Software and Knowledge Architecture. Although the science of Integrated Software and Knowledge Engineering is not offered

in the colleges and universities, these skills are quickly becoming dominant in development arena. Find more on these skills here: http://itofthefuture.com/BASE/Lookup?action=content&issueID=183 [3].

The biggest benefit of online training is still there, students can study at their convenient time with their personal pace, while working under guidance of an experience instructor, who cares about the final destination for a student, - getting a job as a developer, consultant, or a start-up entrepreneur.

Critical Thinking and Communications skills help developers to clearly express their ideas.

Growing importance of communications skills is gradually changing demography of a development crowd. Women, who naturally communicate more than men, have an advantage here.

Find more about the upcoming changes in the article about Women and Men in IT and management: http://WomenAndMen.us [4].

One can think that new tools and approaches, which appear almost daily, solve the problem of transitioning current complexity of enterprise systems to a uniformed simplicity of Semantic Cloud Architecture.

The book "IT of the future: Big Data, Cognitive Computing, and Semantic Cloud Architecture", http://ITofTheFuture.com [5], describes practical steps in transforming enterprise.

The book explains necessity for a common playground, Business Architecture Sandbox for Enterprise (BASE), where Subject Matter Experts and Developers can try new methods and immediately apply them to current business problems. BASE allows a business analyst collaborate with a developer on describing, designing, and implementing a business process and workflow, while being engaged in a conversation with a computer program.

One can start with a business description of an idea and continue walking over the major milestones on the highway, called Software Development Life Cycle (SDLC).

What are the main business problems we are trying to solve?

I think we all know about enterprise challenges: enormous complexity of hardware and software is conflicting with business desire to cut IT budget … and still add more features and capabilities.

I was naïve, thinking that the tools can solve the problem. The missing component was the skill of Knowledge Architecture. But I jumped too far. Let me step back.

Let me start with the current enterprise challenges and solutions as I can see them. Solutions that can do magic in such different worlds as corporation and education. Solutions that can save IT budget while opening new amazing capabilities.

To clearly distinguish this from science-fiction, and I will talk about the practical use cases.

Every use case was a real problem I struggled with, fell in love, and went to such extreme that I prototyped working systems and patented some of them.

Knowledge-Driven Architecture [6] is the way of architecting systems based on business rules and scenarios. Today we can see new software frameworks, such as Google Robot Framework or Cucumber, that also move in this direction and help Cutting Extra Corners in Software Development Life Cycle.

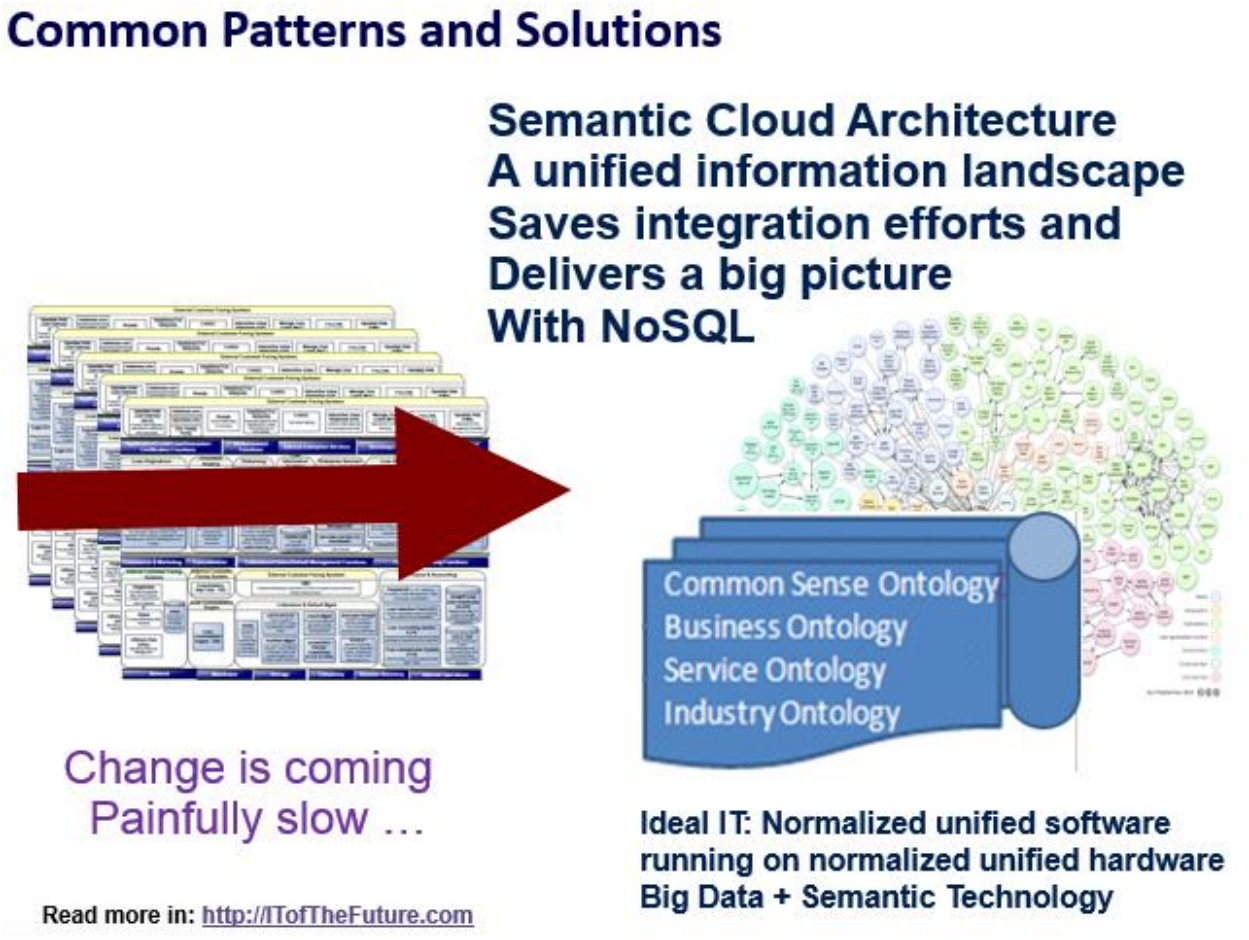
I will consider some specific use cases that became the systems. The names pretty much describe the main purpose and I will talk about details later:

- **Capturing "Tribal Knowledge"** with Rules Collector system [7] and effectively collecting "know-how" in a Corporate Knowledge Factory. No brain surgery is performed here.
- **Adaptive Robot Systems [8]**, which can learn new skills on-the-go. Of course, they need good teachers up to the point they can teach us. And this is coming next.
- **Conversational Design, Modeling, and Manufacturing** [9] – the next step in the development evolution with the Conversational Semantic Service Map, a system to facilitate collaboration of SMEs and developers with the conversational computer program in designing and production of services and products.
- **Changing Formula of Education** and offering a valid alternative to enormously expensive schools. I will start with educational methods and plan to finish with the robot - teachers.

A common pattern of enterprise simplification is moving to a canonical architecture. Today each business application focuses on its own set of data and in most cases creates a specific software architecture to process these data.

Then, big companies spend tremendous efforts and budget to integrate hundred applications and present "a bigger picture of enterprise" to business stakeholders.

An alternative is from the beginning present all data in a Semantic Cloud Architecture, a unified information landscape, which business can query in many expected and unexpected ways.



This is the complete replacement of RDBMS applications. Instead we will use a unified information landscape based on NoSQL.

The benefits are obvious: saving tremendous integration efforts while delivering to business a bigger picture they are so hungry to see and understand.

Current move to a cloud allows companies cut some budget on hardware infrastructure, but real saving starts when we stop bringing to a cloud existing complexity of software packages.

Normalized unified software running on normalized unified hardware! This is the ideal IT environment, inexpensive and efficient.

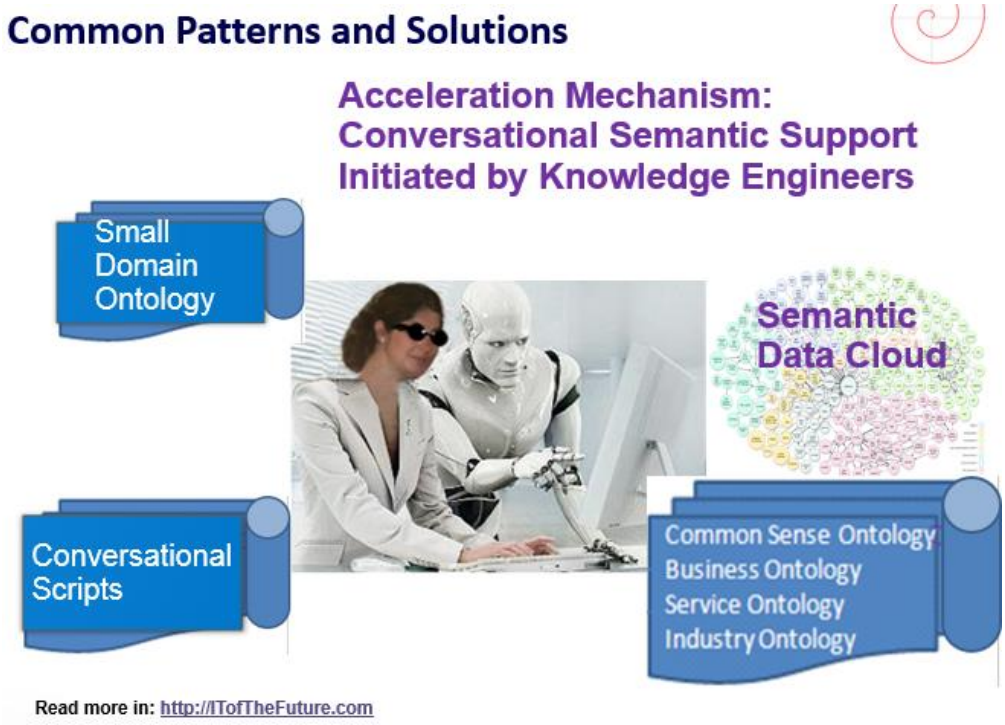
Yes, it is far from traditional applications, where every query is defined by an RDBMS schema and all services are built around these queries. No SQL means no schema, but new mechanisms of retrieving and updating data, new opportunities for distributed and parallel processing.

So far, we talked about Big Data features. Semantic technology integrated with Big Data solve even a bigger problem of understanding incoming information streams, not just crunching more numbers.

Industry is moving in this direction, we can see the changes, although coming painfully slow.

On the top of existing challenges I recently have learned a sad news. We do not have indefinite time. Science research suggests (I am not necessary believe in that) that our universe is dying. We only have several millions years to make things right.

We need some acceleration mechanism. The acceleration mechanism is a conversation between a SME and a computer program, a conversation that is initiated by a Knowledge Engineer, which also is a SME, solving a specific task in her or his business domain. Initial conversational scripts that support the conversation will help solving a problem. In the process they will also help a small initial ontology to gradually grow getting bigger and more powerful with each conversation.



Sometime ago I thought that any SME can perform this role, if there is a tool to facilitate such a conversation.

BASE was designed to be such a tool. But I was wrong. It did not work this way.

I had a training at Cycorp and in 2002 received Knowledge Engineering certificate by Cyc Corporation. Things that looked obvious and natural to me, was not as easy for SMEs without that skills.

Now, let us take a look at the specific use cases.

Cutting extra corners in SDLC. We do this under the umbrella of Knowledge-driven architecture.

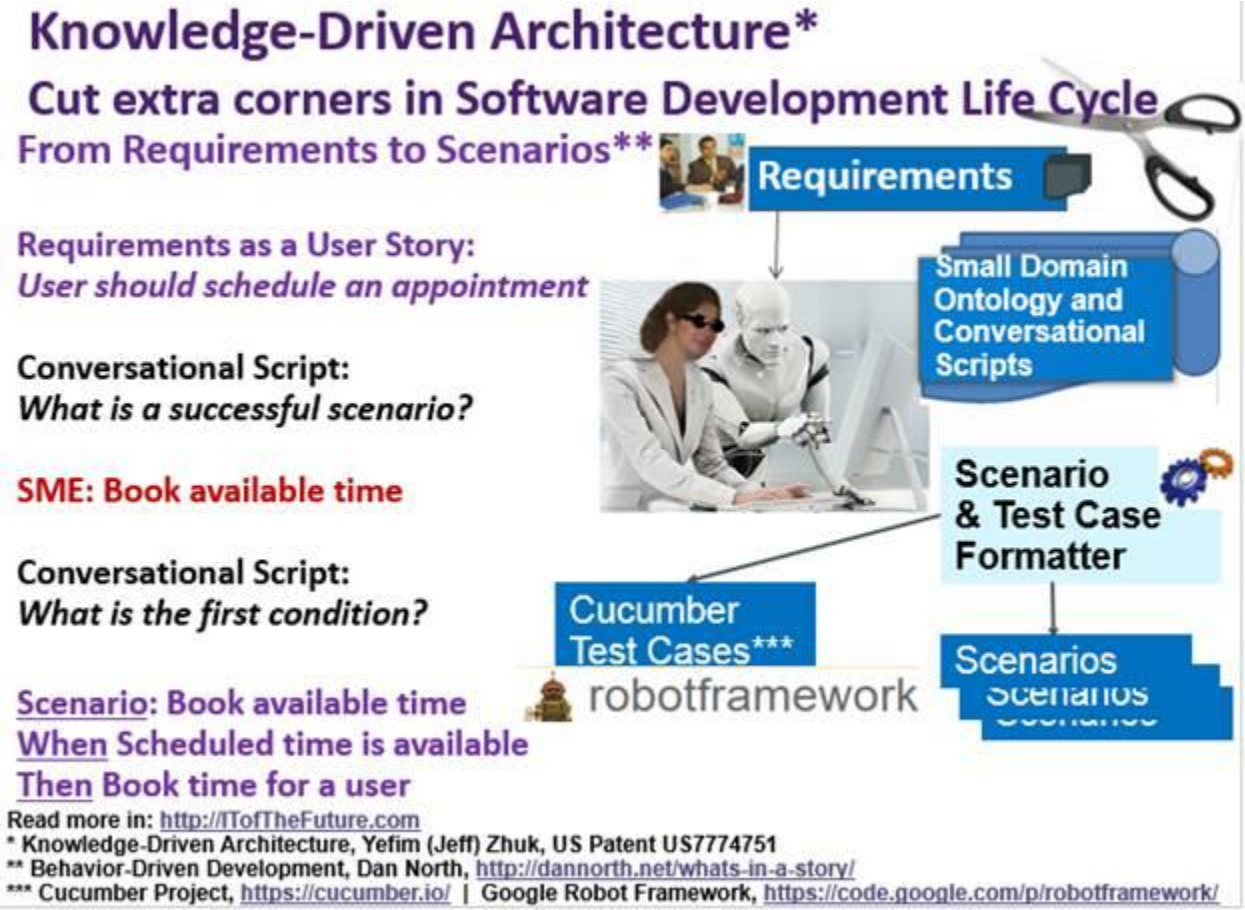
The ideology is simple: move focus from code to business rules and scenarios and provide a smooth and mostly automatic transition to formatted rules and code.

Google Robot as well as Cucumber frameworks partially perform that task. At least one element of that task.

Both frameworks expect a developer read business requirements and transform these requirements into a set of scenarios, using a specific language. Then, based on the scenarios, a developer creates a set of

methods with the test cases for each scenario. This methodology is known as Test-driven development (TDD) and Behavior-driven development (BDD).

With extra help from Semantic Technology, it is possible to expand the advantage by doing a bit more and really cutting extra corners in the Software Development Lifecycle (SDLC) as presented in the illustration below.



Maybe it is time to say something meaningful about these strange partners above. This is about conversational collaboration between a computer program or a robot and a SME.

Do not expect that a computer program will always understand a SME. Opposite, unresolved cases would initiate script-generated questions. These questions help a SME to use proper terms and also gradually improve resolution facilities. The system is automatically getting smarter with more conversations by learning from a SME and constantly increasing available ontology.

With the help of a small initial ontology and initial conversational scripts, a business analyst (SME) will write requirements in the terms of scenarios and rules.

A program can immediately translate this language in the coded test cases. Only at this stage developers comes to the picture to fill out the template methods with specific code.

Time saving is not only on the small development procedure. This process helps business analyst to more precisely express requirements, saving time of unnecessary discussions and "lost in translation" cases.

Semantic technologies allow us to accelerate development by bridging the gap between a User Story and Scenarios with related Google Robot or Cucumber Test Cases. This transformation is performed by the Scenario and Test Case Formatter based on the User Story.

Of course, the whole process must be guided by a conversational script, which helps a developer writing a User Story in no-nonsense terms. Then the conversation is expanded to describe scenarios, which are directly converted to Cucumber Test Case classes.

While the opportunity is there, the reality is less promising. Current enterprise culture misses a whole layer of Knowledge Architecture education, which is necessary here to convert this opportunity into a successful operational process.

Another Use Case is about Validation of Business and Development Models against Requirements, Policies & Regulations.

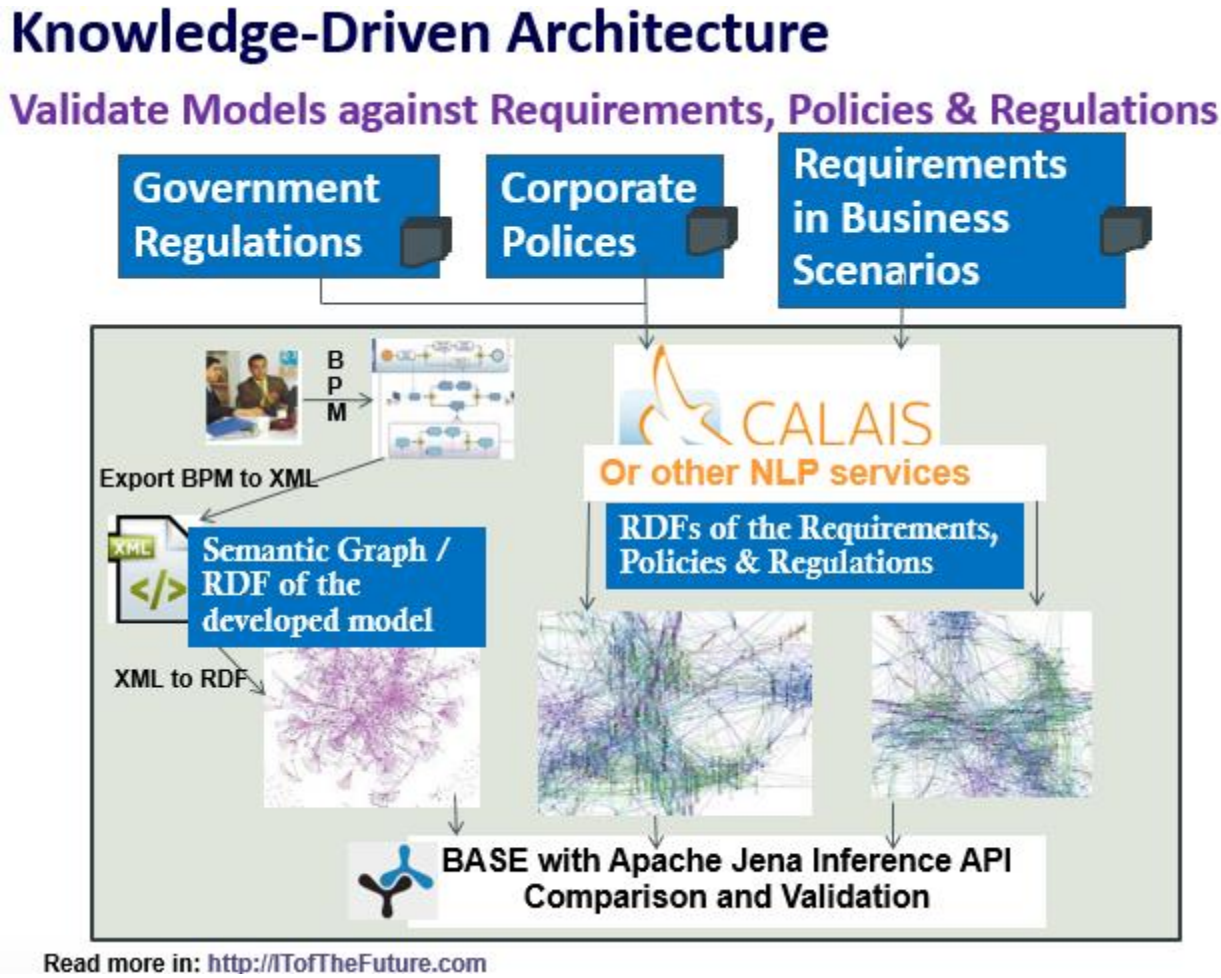
If amount of data is quickly growing, do they make more sense?

Nope! The second law of thermodynamics is very clear about this: we must constantly increase our efforts just to maintain status quo!

"Lost in translation" is a very common use case in software development. Creating the model is one of the first phases of the translation of the requirements to a working application. Verification or testing is one of the latest phases, when each "lost and found" item is extremely expensive.

When several hundred pages of requirements are extended by many other documents with industry standards and regulations, it is close to impossible to verify if the model is compliant… unless we start using the magic of semantic technologies.

Computerized help is highly appreciated here. We do not expect "everything" done by a program but a lot of subjects in the requirements can be automatically verified and others can be displayed as red flags for manual verification.



It appears that one of the biggest giants in financial news, Thomson Reuters, offers its help in tagging the world. Several years ago the company launched a free service, "Open Calais", which adds semantic metadata to unstructured text or HTML documents, producing RDF.

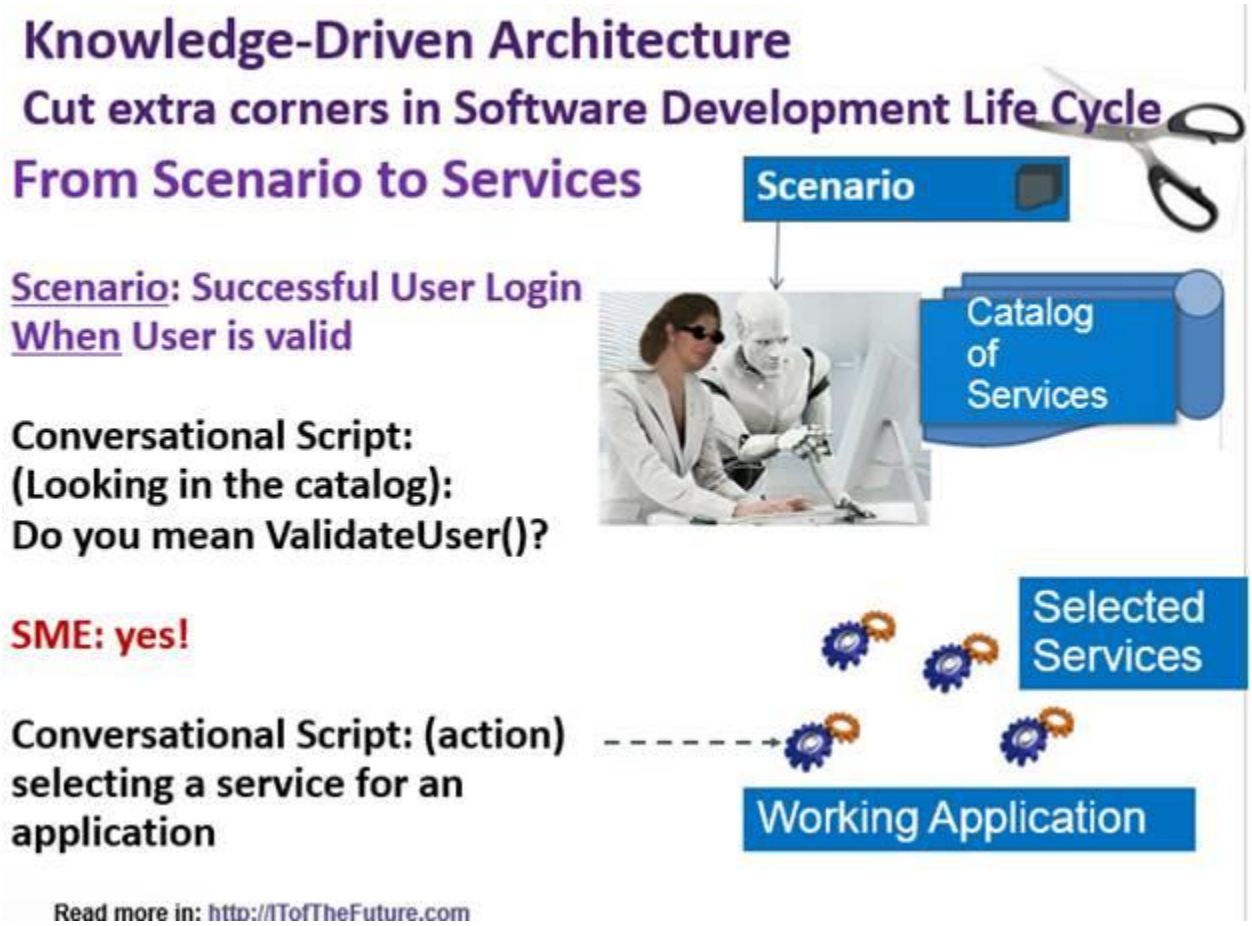
On the left side of the picture, the architects create a model. In this example it is a business process model created with the Business Process Modeling (BPM) tool.

On the right side of the diagram, the same requirements and other documents are going through the **text to RDF** conversion to become another semantic graph in the RDF format.

At the bottom of the diagram, the inference engine by Apache Jena or another provider compares both semantic graphs and makes a conclusion on matches and mismatches of the models, based on the rules created for the task.

Another case is another step in SDLC: connecting scenarios to services.

From the beginning of this century, I promoted SOA with the belief that created services will be connected by business into applications. SOA did not fulfill this promise. Having thousands of services does not help. Business does not want to deal with the huge catalogs written in special terms. Even developers have hard time to map the services to required business functions.



Conversational semantic support allows a SME transition from business Scenario to Services by mapping business language to catalog terms and asking questions to bridge business language to service terms.

Scenario: Successful User Login

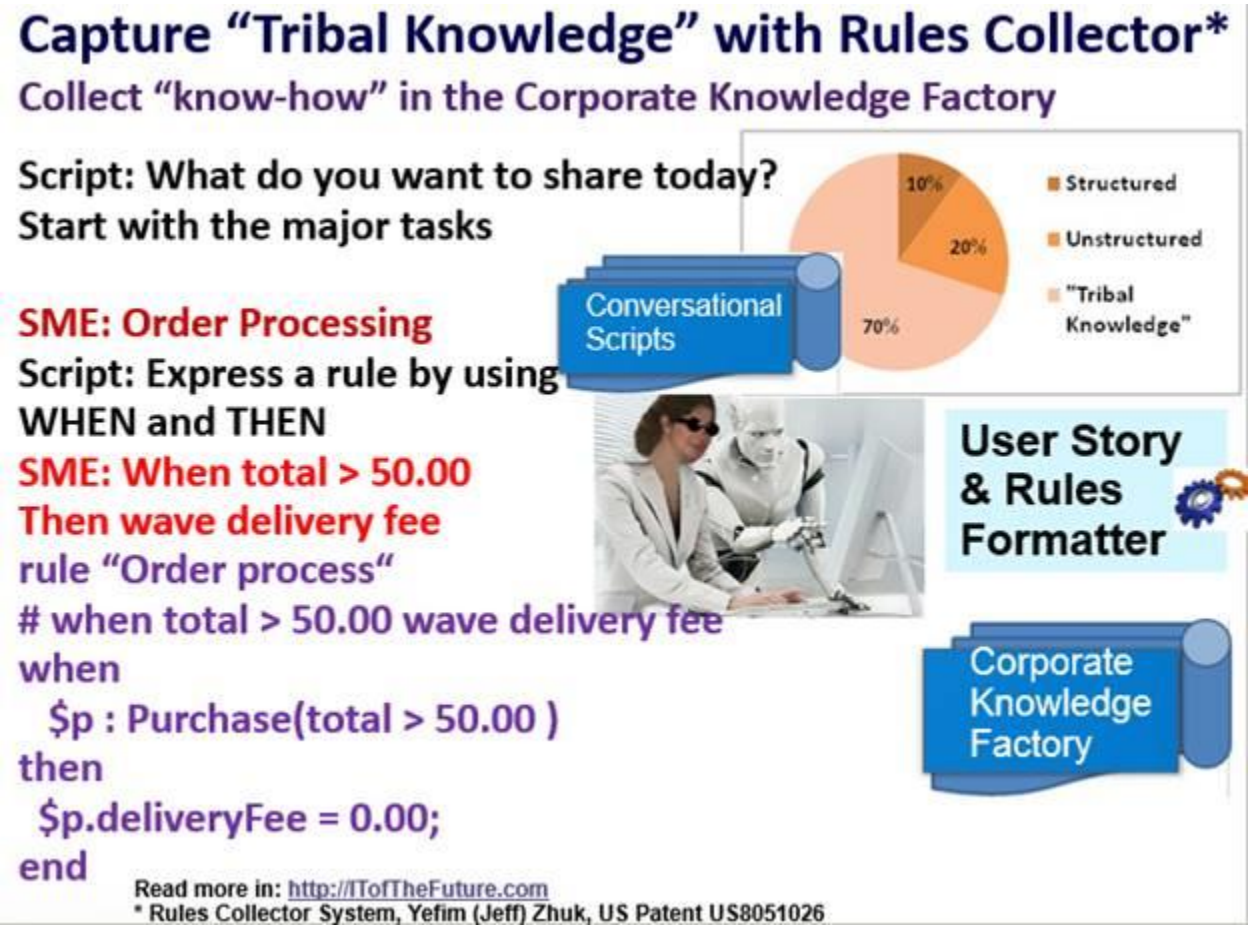
Business Analyst is writing a sentence: "When User is valid…"

Conversational Script (Looking in the catalog): Did you mean to use the ValidateUser() service?

Business Analyst: yes! **Conversational Script**: (action) selects the ValidateUser() service.

Corporate knowledge or "know how" can be split into three categories: structured data – in relational databases, unstructured data – text documents in folders and web sites, and the biggest portion of information that is used daily in business routine and has never been captured. It is so-called "Tribal Knowledge". My conservative estimate of the ratio between structured, unstructured and "tribal" knowledge is 10%, 20% and 70%.

By retiring "baby boomers" or replacing "experienced and expensive" with "young and cheap" corporations actively lose huge portions of tribal knowledge.



The Rules Collector [7] is a patented system, which can converse with a SME to retrieve and translates "tribal knowledge" into the rules and scenarios.

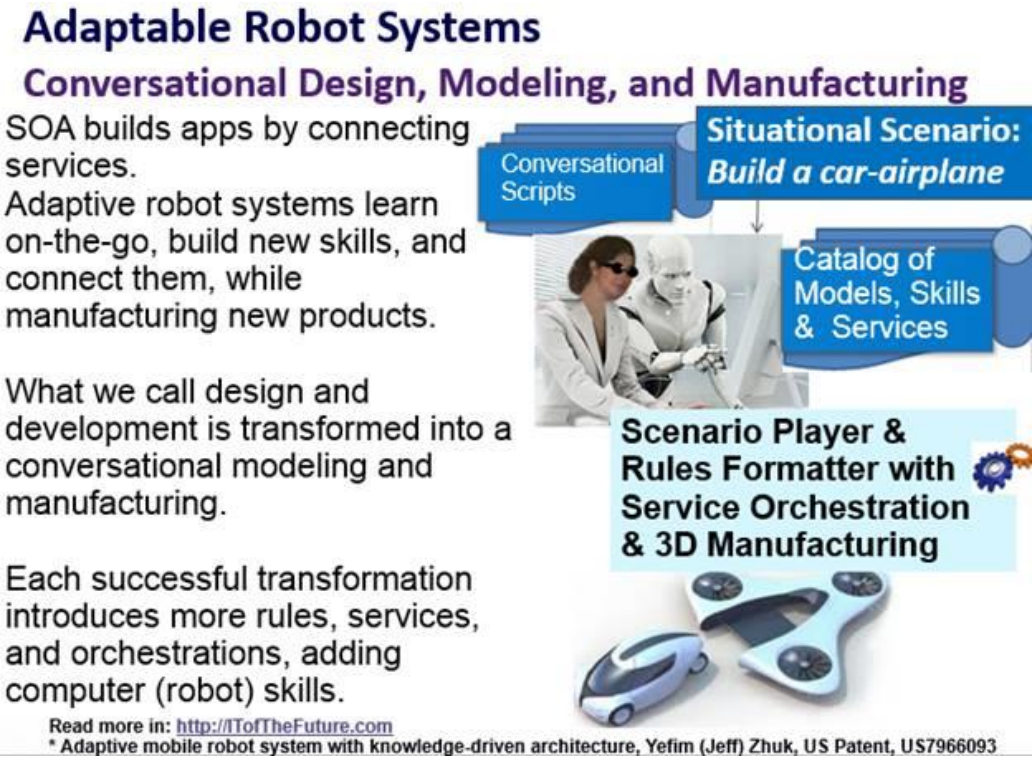
The conversational approach to knowledge acquisition combines the power of Big Data and Semantic Technologies with the human intuition.

The next use case expands on SOA promise to provide subject matter experts with the ability building software applications by connecting services.

Just imagine a **Conversational Semantic Service Map** [8], a system for collaborative design, assembly on-the-fly, execution, benchmarking, and negotiation of computer services and applications, a common destination for developers and subject matter experts.

And this was just the starting point for a much more ambitious use case, which integrates software and knowledge engineering with robotics technology. to improve robot-to-robot and robot-to-human conversational interface and provide on-the-fly translations of situational requirements into adaptive behavior models and further down to service scenarios for a collaborative robot teams.

Adaptive robot systems [8] can learn on-the-go and build new skills, while providing on-the-fly translations of situational requirements into adaptive behavior models and further down to service scenarios for a collaborative robot teams.



Each successful transformation introduces more rules, services, and orchestrations, adding computer (robot) skills. We can say "Build a car that can fly" and a robot will look into existing catalog of skills, and if it is not there comes back to you and to the network of other robots for more details.

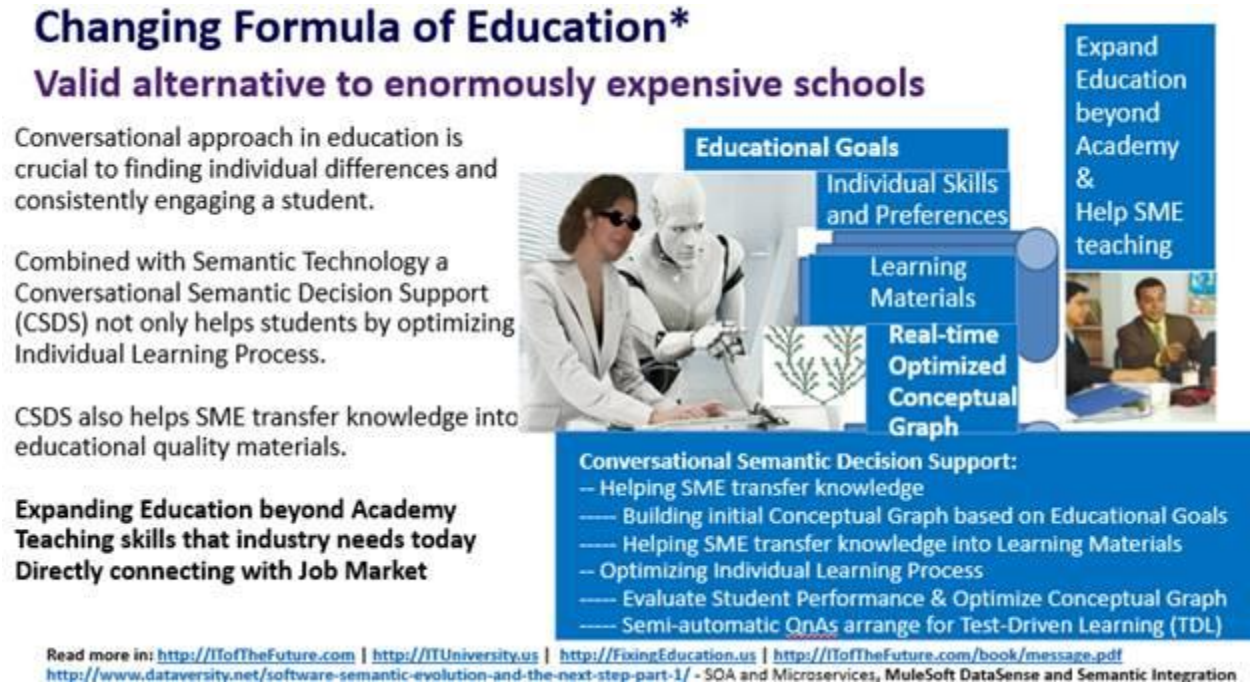
Can Semantic Technology apply to Education? Yes! We change the formula of education!

Current Formula of Education:

- Colleges and Universities: the main channel to access education
- Curriculum: several years behind industry demands
- Graduates: have hard time finding work

Internet Technology University (ITU) and Semantic Technology change this formula.

ITU uses Smart Cloud services with Conversational Semantic Decision support (CSDS) to create a new educational platform. CSDS will partner with teachers to help finding individual differences in student's learning style, providing immediate feedback and consistently engaging a student.



Even more important is help provided to instructors. CSDS will help SME building Conceptual Graph and transfer knowledge into educational quality materials. We expand education beyond Academy, beyond colleges and universities, helping SME, who wants to share, become SME-instructor, teaching skills that are needed today and tomorrow, directly connecting students with Job Market.

Some companies, such as IBM, Google, Facebook, already started this process. CSDS makes it efficient.

See http://ITUniverity.us | http://FixingEducation.us | http://itofthefuture.com

What can make these practical dream - cases reality?

Coming back to Earth, I have to admit that an essential component is missing today.

Someday every college and university will teach integrated software and knowledge engineering, similar to the curriculum created by Internet Technology University, http://ITUniversity.us

These are currently missing development qualities, which allow developers to perform SME/Knowledge Engineering Tasks:

- **Verify and Correct Domain Concepts after automatic Text Extraction**

- **Verify and Correct Domain Ontology**
- **Verify and Correct Conversational Scripts Generated with a Domain Ontology**
- **Verify and Correct Transformations of Requirements into Scenarios and Rules**

A brief summary and another attempt to answer the original question:

What are the most important development skills?

We used to talk about **programming languages** and tools.

Then, **service-oriented architecture** forced developers to pay closer attention to a **Business Domain**. A good developer must understand business goals and models.

Semantic Technologies disrupt and accelerate development. But technology itself does not do the magic.

No matter what tools you have, **Knowledge Engineering** skills is necessity for success. Companies start from borrowing the skills from consultants, then learning and growing locally to expand and improve efficiency of their core business.

The closest known development role is a Business Architect. But, I think, we are looking for a bigger mix that includes a Knowledge Architect and Business Architect all together and adds much more efficiency to the development process.

In the nearest future computer programs will help us in so many areas that we might need that robot -partner walking around. Army of specialized robots work today at car factories. More universal models will be able to converse with us. Starting with simple conversational scripts, created by us, they will be able to quickly learn and help us transforming our ideas into 3D models and products.

Start learning today with ITU, get ready to join the challenging and exciting work.

Have a question, an opinion or comments to share? Welcome to the discussion board!

References:

1. Software Semantic Evolution, Jeff Zhuk, From "all-in'one" programs to SOA and Microservices, to RAML and DataSense by MuleSoft and the next step,

   http://www.dataversity.net/software-semantic-evolution-and-the-next-step-part-1/

2. Fixing Education and Corporation in one shot, an ambitious goal, but a doable project, http://FixingEducation.us.

3. Critical Thinking as the pre-requisite to Integrated Software and Knowledge Architecture, http://itofthefuture.com/BASE/Lookup?action=content&issueID=183.

4. Women and Men in IT and management: http://womenandmen.us/WomenAndMen.pdf

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. Knowledge-Driven Architecture, Yefim Zhuk, Streamlining development and driving applications with business rules & scenarios, US Patent, http://www.google.com/patents/US7774751
7. Rules Collector system, Yefim Zhuk, Transforming "tribal knowledge" into formal rules to drive applications and business processes, US Patent, http://captureknowledge.org/
8. 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
9. Development Factory, The system for conversational development, assembly on-the-fly, execution, benchmarking, and negotiation of complex projects; US Patent, https://patents.google.com/patent/US10956676B2/en