An Introduction to Knowledge Computing

Vagan Terziyan [a], Oleksandr Shevchenko [b], Mariia Golovianko [b]

[a] *Department of Mathematical Information Technology, University of Jyvaskyla, Finland; P.O. Box 35 (Agora), 40014, Jyvaskyla, Finland*

Telephone (GSM): +358-50-373-2127

Fax: +358-14-260-2731

E-mail: vagan.terziyan@jyu.fi

[b] *Department of Artificial Intelligence, Kharkov National University of Radioelectronics, Lenin avenue 14, 61166 Kharkov, Ukraine*

E-mails: shevchenko@sw-expert.com; golovianko@gmail.com

Abstract

This paper deals with the challenges related to self-management and evolution of massive knowledge collections. We can assume that a self-managed knowledge graph needs a kind of a hybrid of: an explicit declarative self-knowledge (as knowledge about own properties and capabilities) and an explicit procedural self-knowledge (as knowledge on how to utilize own properties and the capabilities for the self-management). We offer an extension to a traditional RDF model of describing knowledge graphs according to the Semantic Web standards so that it will also allow to a knowledge entity to autonomously perform or query from remote services different computational executions needed. We also introduce the concepts of executable knowledge and knowledge computing on the basis of adding an executable property to traditionally used (datatype and object) properties within the RDF model. The knowledge represented with such an extended model we call as an *executable knowledge*, or the one which contains explicit (executable) instructions on how to manage itself. The appropriate process of the executable knowledge (self-) management we call as a *Knowledge Computing*. Unlike the knowledge answering machines, where computations over knowledge are used just for addressing a user query, the knowledge computing in addition provides computations for various self-management purposes. The paper also presents some pilot (proof-of-concept) implementation of the executable knowledge as a plug-in to Protégé ontology development environment.

Keywords: self-managed systems; knowledge; RDF-graph; knowledge management; Semantic Web; knowledge ecosystems; executable knowledge; knowledge computing; knowledge processor

1. **Introduction**

Our information society [1-2] is rapidly transitioning to the knowledge economy [3], in which knowledge management is transforming to the knowledge ecosystem model [4] and various suppliers are converging from the old to the new media [5].

Current technological world is a pragmatic one: we look at any world entity (a living, artificial, abstract one) through smart applications and ask a simple question: how can I utilize it?), i.e., consider “Everything as a Capability”. A capability is usually provided explicitly or implicitly through a product or a service. Some entities may not provide useful capabilities directly but can facilitate capabilities of other products, services or users. Knowledge is a facilitator of many important capabilities, such as decision-making and learning. Knowledge is converted into economic value by processes that involve interconnected cognitive, physical or social actions in physical or virtual spaces as a result of economic or communicative transactions to achieve some organizational goals. Unlike most real world resources depleted after being used, knowledge can be shared and reused, it grows through applications and enhance its inherent value within knowledge storages. Knowledge economy [6] concerns exchanging knowledge-based products and services within knowledge markets [3], which are based on mechanism enabling, supporting, and facilitating mobilization, sharing, or exchanging information and knowledge among providers and users. The New Media [5] increases production and distribution of knowledge as a collective intelligence effort (with online interactions between users and producers) to make the existing knowledge easier to access and reuse.

Knowledge collected from different heterogeneous sources needs standardization to enable “understanding”, which is the process of connecting (linking) new information and knowledge to the one already stored. Currently popular term “Linked Data” [7, 8] may therefore have one simple meaning: “data which has been understood”. And when data is linked then one can make a deeper, more intelligent and efficient analysis of it (mining, knowledge discovery, pattern recognition, diagnostics, prediction, etc.). It is known that something represented as a Linked Data (based on RDF, OWL and other standards) can be relatively easily linked with other public data sets for creation of a cloud of linked open data (the Semantic Web) [9]. There are many collaborative data-linking efforts mashing up features from Web 2.0 and the Semantic Web [10], e.g., Linking Open Data (w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData), DBpedia (dbpedia.org), Freebase (freebase.com), Factual (factual.com), INSEMTIVES (insemtives.eu/index.php) and many others, which provide structured content published and driven by volunteers. Collaborative creation and evolutionary maintenance can be used not only for Linked Data itself (as a knowledge graph) but even for ontologies ("folksonomies" or social tagging in this case).

Another trend is related to making knowledge computable. See, e.g., the effort of Wolfram|Alpha (wolframalpha.com) which is a query-answering tool providing visualization capability (the authors call it a "computational knowledge engine for the Web"), which allows dynamic computations on top of structured data (aka declarative knowledge) due to various procedural attachments and built-ins within it. The challenge however would be to make such computational knowledge enabled for Semantic Web, i.e., suitable not only for humans but also for various heterogeneous and potentially interoperable applications, where a group of knowledge-driven computations consumers will include other external computational processes of remote systems, which have to "understand" what they query and what answers they get.

Knowledge has to be well managed in order to be capable for effective computations. We have to admit that knowledge is such a complex and dynamic entity that it would be naïve to assume that the traditional knowledge management approaches would be enough to meet all the emerging challenges. We require self-managed evolutionary knowledge, which will be able to manage itself autonomously and therefore will have some extra knowledge about its structure and possible behavior. This is a new context for considerations for knowledge and knowledge-based systems. Knowledge provision (or/and computations or inferences on top of it) has always been the only concern of the systems. However now an emerging concern is how to do the same for needs related to knowledge self-management and autonomic evolution. The concept of a knowledge ecosystem is appeared to be a suitable one to refer to a knowledge management on enabling self-organization in response to changing environments based on the dynamic evolution [11, 4].

In this paper we are going to address the challenges related to knowledge self-management. We offer an extension of the semantics of the traditional RDF model used for operation with knowledge graphs in order to enhance autonomous behavior of knowledge with respect to the Semantic Web standards. We call knowledge based on the extended model *executable*, which means that it contains explicit (executable) instructions on how to manage itself (i.e. self-management enabled). We call the correspondent process of the executable knowledge (self-)management *Executable Knowledge Computing*. It serves for a wide range of self-management purposes to accommodate the changes happening in the Linked Open Data which is much wider than just user`s queries addressing performed by Google Knowledge Graph (google.com/insidesearch/features/search/ knowledge.html) or the Wolfram|Alpha (wolframalpha.com).

The rest of the paper is organized in the following way: we continue the introduction and discuss current trends in Information and Communication Technologies figuring out the needs for autonomic and self-managed solutions related to knowledge economy in Section 2; in Section 3 we provide some basic characteristics and requirements to appropriate self-managed systems; we discuss what kind of knowledge would be needed for such self-managed systems in Section 4; in Section 5 we discuss combination of the conflicting Open World and Closed World assumptions to enable self-management in knowledge-based systems; existing approaches on enabling procedural attachments into declarative knowledge is briefly described in Section 6; Semantic Web services and various approaches around this concept to add semantics to a procedural knowledge are described in Section 7; in Section 8 we introduce the concepts of executable knowledge and executable knowledge computing utilizing the traditional RDF model extended by a new property type; we present some pilot (proof-of-concept) implementation of the executable knowledge and a reasoned for it as a plug-in to Protégé ontology development environment in Section 9; and we conclude in Section 10.

2. Current Crisis and New Trends in Information Technologies

Analysis of the current crisis (status of related markets and employments) of the ICT (Information and Communication Technologies) domain [12] indicates certain trends, which, e.g., demonstrate negative dynamics related to ɪ**C**T (communication technologies) component and probable recovery and growth for the **I**CT (information technologies) component of the ICT.

It is well known that “Information Technology is concerned with technology to treat information towards user needs”. We have 3 major keywords here (*information, technology* and *user*) and with each of them the new challenging trends are associated as follows:

Information (data, knowledge) is becoming huge (in amounts and dimensions), globally distributed (by location), heterogeneous (by the nature of the source), dynamic (changing with space-time and other contexts), multi-disciplinary (by scope) and is already beyond our capability to successfully process, store and understand it with existing tools. Many experts consider that this Big Data challenge has been one of the most exciting opportunities in the past 10 years [13].

Technology provides capabilities (in forms of applications (as products or as services with more shift towards services)) to manipulate Information. The capabilities are also becoming huge (by number and complexity), globally distributed (by location), heterogeneous (by the nature of the developer or provider), dynamic (configurable adapting to the space-time change and other contexts) and multidisciplinary (by scope). Emerging service industry (so called Web-based service economy) on top of the Internet of Services with global service delivery platforms, utilizes and expands Web 2.0 and future network infrastructure (Internet of Things). It is promoted by giants like SAP, Amazon, eBay, Google, Siemens, Philips, etc. According to the SAP vision, the Web-based service economy in the Internet of Services will likely be an integral part of the future economic innovation, value creation, growth, and employment [14].

User is also becoming "huge" (in number and social interconnections), globally distributed (by location), heterogeneous (by nature ("Everything-as-a-User" [15]) and experience), dynamic (profiles and preferences are space-time and other contexts dependent) and multidisciplinary (by scope of interest). Today user is also an active data and knowledge contributor and capability provider through the Web.

The major requirements to a possible solution to meet all these challenges are:

Self-management (self-configuration, self-optimization, self-protection, self-healing, etc., features of autonomic computing) is needed to handle huge scale complexity (volumes of information, numbers of technology capabilities, variety of users and all related interactions);

Semantics (according to Semantic (Web) Technology) is needed to enable self-management and to handle heterogeneity of information, technology capabilities and users;

Smart Integration (of information, capabilities, or users) is needed to enable interconnection (e.g., Linked Data) and interoperability among all "actors" and "components" and to enable seamless and automated compilation of new complex systems from available components;

Context-Awareness (including context modeling and computing) is needed to handle dynamic aspect of current IT trend (e.g., word "mobile" has now wider meaning, like "changing in context");

Architectures related to *SOA and Cloud Computing* may serve as technological and business ecosystems for multidisciplinary domains.

Therefore: the major emergent topics around Information Technology that are addressing the current IT challenges would be:

- Self-management;
- Big and Linked Data, Semantics, Interoperability and Integration;
- Service-Oriented Architectures and Cloud Computing with the enhanced general slogan: "*Everything-as-a-Service-for-Everything*!"

The three major aspects of such enhanced vision of "*Everything-as-a-Service Engineering*" would be:

1. *Everything as a Service Provider*. This traditional concern sounds like: What (infrastructure, platforms, software, interfaces, data, etc.) should be additionally provided to make some product or system capable of performing its functionality (data or capability) as a service for external users, businesses or systems through the Web?

2. *Everything as a Service Consumer*. Here we have much more challenging question: How to design products and systems so that they will be capable of automatic real-time discovery, query and utilization of external data and capabilities for better meeting their design objectives and beyond?

3. *Everything as a Self-Service*. Major research question here is: How to make systems self-aware, context-aware and capable of self-configuration, self-optimization, self-protection and self-healing while adapting their design objectives in real time to changing execution environments according to the "Open World assumption" (i.e., a system should be able to handle new situations, which were not known during its design time).

Not only technologies are rapidly changing but also the requirements towards future IT workers and their skills (which causes new challenges for the universities) are also changing radically. Traditional requirements for the open positions in IT contained before the list of concrete topics and products a candidate was expected to know or be skillful in. Now the requirements may sound shorter and more challenging, like, e.g., "people need to think beyond the routine, and need to have the ability not just to adapt to change, but to help create it" [16].

We have observed recently a dramatic increase in our ability to collect data from various sensors, devices, in different formats, from different users, applications and services. The amounts of these data are already so huge that it is beyond our capability to successfully process, store and understand it. The major challenge would be to find balance between two evident statements: (a) the more data you have, the more potentially useful patterns it may include; (b) the more data you have, the less hope that any sophisticated machine-learning algorithm will be capable to discover these patterns in the acceptable time frame.

Knowledge (as the result of some analytical processing over data) definitely has more value than the data from which it has been originated and we may expect that big volumes of processed data will result to big volumes of produced knowledge (which is not always true but likely to happen in many cases). Knowledge has additional challenges of being “big” including challenge of resolving contradictory (and evolving) opinions of everyone on everything where managing the authority and reputation of “experts” will play an important role [17]. The Semantic Technology (RDF, OWL, etc.) was originated aiming to break down the data silos and at the same time to enable efficient (big) knowledge management. However taking into account the trends mentioned above one may expect that the (big) knowledge to be effectively managed as a complex system should be a proactive, self-managing, self-evolutionary entity capable of consuming and providing services and self-services over the Web. Such environments, in which knowledge can “live” autonomously are known as a knowledge ecosystems (see, e.g., [18]), are considered as a kind of digital ecosystem (alternative to the traditional knowledge management approach with its directive management) towards enabling self-organization and dynamic evolution of knowledge interaction between entities (interlinked knowledge resources, databases, human experts, and artificial knowledge agents) in response to changing environments.

The question however is whether the current standards for knowledge representation and sharing are suitable to enable its emergent self-management capabilities and, if not, then what might be the needed update? This is the issue of this paper.

3. Autonomic Computing, Self-Management and Evolution

Started by IBM in 2001, the Autonomic Computing refers to the self-managing characteristics of complex computing systems to manage themselves without direct human intervention (i.e., the human defines general policies that constrain the self-management process). According to IBM, the major four functional areas of autonomic computing would be: self-configuration (automatic configuration of system components); self-optimization (automatic monitoring and ensuring the optimal functioning of the system within defined requirements); self-protection (automatic identification and protection from security threats); and self-healing (automatic fault discovery and correction). Other important capabilities of autonomic systems are: self-awareness (capable of knowing itself); self-adaptation (acting accordingly to own environment and surrounding context observed); being non-proprietary (function in a heterogeneous word of open standards); and anticipatory (automatically anticipate needed resources) [19].

According to [20] a self-managed system must be able to dynamically change its behavior at runtime following user requirements, execution environments, or technologies change, therefore the system manages itself given high-level objectives. Such systems are based not only on reusable (software) components but rather on dynamically reconfigurable ones. Therefore a system reconfigures itself (either separate components or their communication logic) to: (a) address such changing objectives (expectations, requirements) from the user, which were not anticipated at the design phase; (b) address such changes in the execution environment, which were not anticipated at the design phase; (c) address such changes in the technology, which means radically new utilization context (communication networks, devices, standards, etc.) for the system.

Is the list above of self-managing characteristics would be enough? Probably not. The self-managed systems are naturally proactive, which means that they are capable not only to adapt themselves to the environmental change but also create changes in the environments, i.e., adapting the environments to own benefits when possible and appropriate. According to [21] a self-managed system is an autonomous system like a robot is. This actually means that it is proactively adaptive to the environmental changes, not just reactive. A software architecture of self-managed system is one in which components automatically configure their interaction to be compatible with an overall architectural specification and the objectives of the system. One of such major objectives applied to self-managed systems is to minimize the degree of explicit management necessary for (re)construction and subsequent evolution whilst preserving the architectural properties implied by its specification [21]. Challenges here were noticed as follows: reconfiguration of the software components, which ensures application consistency; decentralized configuration management, which can tolerate inconsistent views of the system state, but still converge to a satisfactory stable state; on-line (perhaps constraint based) planning for the goal management layer.

Consider appropriate definition of an intelligent agent concept from [22], which fits well these extended requirements to the self-managed systems. The definition is based on the concept of Semantic Balance [23] between internal (own configuration) and external (outside world configuration) environments of an agent as a kind of “survival instinct”.

An Intelligent Agent is considered in [22] as such proactively-adaptive self-managed entity that is expected (for survival) to *keep continuously balance between configurations of its internal and external environments* in such a way that in the case of unbalance agent can choose the behavioral option from the following list:

- *make a change within the configuration of the external environment* to be in balance with the internal one;
- *make a change within the configuration of the internal environment* to be in balance with the external one;
- find out and *move to another place* within the external environment, which configuration is such that needed balance occurs without any changes;
- *communicate and collaborate* with one or more other agents (human or artificial) to be able *to create a community*, which collaborative internal environment and its configuration will be in balance with the external one.

The concept of an agent fits well the expectations from the self-managed systems and therefore it would be reasonable to consider the extensive self-managed systems as the agent-driven ones. For the further needs of this paper let us fix some characteristics of a self-managed system. A self-managed system should be capable: (1) to observe (and record) its own configuration and current state including configuration and behavior of the management engine on top of it through *internal sensors*; (2) to observe (and record) the state and the behavior (including inquiries) of the outside world entities through *external sensors and communication channels*; (3) to reason (discover) the need and *objectives for self-reconfiguration* by smart comparison of the records from (1) and (2); (4) to create a *self-reconfiguration plan* based on the objectives from (3); (5) to execute appropriate actions from the self-reconfiguration plans from (4) through *internal effectors*; (6) to proactively order, query, or execute the needed reconfiguration actions from the external world entities (if the self-reconfiguration is not possible) through the *external effectors and communication channels*.

Such adaptive, proactive, mobile, collaborative and context-aware systems capable to dynamically change their behavior at runtime are difficult to create. Dynamic reconfiguration of such systems can generate inconsistencies, integrity problems and combinatorial explosion of possible variants, all of which leads to a great complexity, considerable technical challenges and high implementation costs.

Even such a sophisticated self-managed (a kind of “artificial life”) system as described above may meet one day such rapidly evolving circumstances that it will not be capable anymore to reconfigure itself in a real time to address the new challenges, to fit its design objectives and therefore becoming useless for further exploitation and “dies”. Then it usually happens, that the next generation system is created, which may inherit something from previous system but also should have some principally new features. Because of that we admit that it is not always possible to a system to adapt itself to a change within its lifetime. This means that the self-management capability only may not be enough for the system to "survive" and we are coming to the necessity of the (self-)evolution. The nature invented a long-term (beyond single system lifecycle) adaptation instrument named genetic evolution. Evolution is known to be the change in the inherited features of populations (of natural or artificial life forms) over successive generations. Evolutionary processes provide diversity at every level of life organization.

4. Knowledge for Survival

Let us agree with [24] saying: "Someone once said that that there is nothing new in this world, we keep on reinventing the wheel. I have no pretension that this work contains anything new, since I borrowed heavily from the work and insights of others. What is different is how these insights are cobbled together into a different and perhaps new whole" (new system).

In terms of self-management and evolution, knowledge is essential for a living system`s survival. According to [25], knowledge is needed to make a complex unpredictable world understandable for making better decisions within it. It is also believed [26] that knowing things gives evolutionary advantage to those who know better how to adapt rather than die.

The role of knowledge for self-management of individuals can be seen from e.g. BDI (Beliefs-Desires-Intentions) model with various enhancements [27], which is traditionally applied for programming software agents' behaviors. The abstract formula of agent behavior would be: survival as the "basic instinct"; different and changing *desires* (goals and objectives) appear (inferred) according to the survival needs; *intentions* (executable plans for achieving objectives) are inferred based on own *beliefs* (knowledge about itself and the environment).

The role of knowledge for self-management of the groups of individuals is associated with the knowledge management concept. In today's hypercompetitive environment, knowledge management becomes a vital component for modern organizations. Knowledge management relates to an organization's ability to systematically capture, organize, and store information exploring technologies like business intelligence, collaboration, knowledge transfer, knowledge discovery and mapping, etc. [28]. In the autonomic systems context, the knowledge includes essential part of a self-knowledge (as named in [29]) with corresponding management components like self-monitoring, self-learning, self-diagnostics, etc.

An autonomic system should obviously work with so called *dynamic knowledge* (see, e.g., [30]), which is dynamically changing knowledge, providing on-demand, in-context, timely, and relevant information. Representation of dynamic knowledge reacting to any changes in business environment and the user's needs is possible with autonomic ontologies [31, 32]. To support their efficiency important techniques including detecting and fixing broken links in linked data sets [33], monitoring and notifying data changes [34, 35], rebalancing graph structures [31, 32] are introduced. Powerful and expressive tools and languages (such as, e.g., LUPS [36]) are used for representing and proper handling of conflicting updates as addressed in [30].

Is self-knowledge declarative or procedural? A procedural knowledge (or knowledge on how to do something) is known to be a knowledge focused on obtaining a result and exercised in the accomplishment of a task, unlike declarative knowledge (propositional knowledge or knowledge about something) [37]. Procedural knowledge is usually represented as finite-state machine, computer program or a plan. It is often a tacit knowledge, which means that it is difficult to verbalize it and transfer to another person or an agent. The opposite of tacit knowledge is explicit knowledge. We believe that an autonomic system needs a kind of a hybrid of explicit declarative self-knowledge (as knowledge about own properties and capabilities) and explicit procedural self-knowledge (as knowledge on how to utilize own properties and the capabilities for the self-management).

5. Open World vs. Closed World Assumptions

Individual knowledge and knowledge we collectively share in the Web describes only a small portion of the world around us and the larger portion remains unknown. This so called Open World assumption is used as the basis within most of current ontology reasoning support tools. Humans as well as various systems however are making their decisions within Closed World of known facts and therefore these decisions may lead to a failure and will be optimal only if a complete knowledge is available (i.e., almost never).

In the formal logic, the Open World Assumption (OWA) is the assumption that the truth-value of a statement is independent of whether or not it is known by somebody to be true. It is the opposite of the Closed World Assumption (CWA) or "negation as failure", which holds that any statement that is not known to be true is false. For example, if the only knowledge a system has about some John would be: "John has daughter Mary", then according to the CWA it would automatically mean that a statement "John has daughter Suzanna" is false, however according to the OWA the reaction to the same statement “John has daughter Suzanna” would be “I do not know”. Therefore the CWA allows a system to infer, from its lack of knowledge of a statement being true, anything that follows from that statement being false, while the OWA limits those kinds of inference and deductions because of ignorance. Within the OWA-based systems, from the absence of a statement alone, a deductive reasoner cannot (and must not) infer that the statement is false. The OWA reflects the monotonic nature of the first-order logic, i.e., adding new information never falsifies previous conclusions. This fact however limits possibilities of the OWA-based systems to benefit from the non-monotonic reasoning techniques (e.g., default reasoning) where previous conclusions can be invalidated by adding more knowledge.

Semantic Web languages such as OWL assume the OWA while most of procedural programming languages and database management systems assume the CWA. An important question for the emerging Semantic Web is how to best combine description logic-based open world ontology languages, such as the OWL, with closed world non-monotonic logic rules. Ontologies are a standard OWA formalism while rules usually apply the CWA. A combination of ontologies and rules would clearly yield a combination of the OWA and the CWA and this is not only of interest for current applications in the Web, but also as a highly sophisticated means of knowledge representation in general [38]. However, combining rules and ontologies is a nontrivial task, since a naive combination of ontologies and OWA-based rules is known to be undecidable [39]. One of the most solid proposal for reasonable combination is known as hybrid MKNF knowledge bases [40, 38] consisting of ontology axioms and rules, which is based on a well-founded semantics that promises better reasoning efficiency and compatible with both the OWL-based semantics and the traditional Well-Founded Semantics for logic programs. As argued in [40], a hybrid formalism combining rules and ontologies should satisfy certain criteria: (a) faithfulness, i.e., preserving semantics of both formalisms; (b) tightness: i.e., both the ontological description logic component and the rule component should be able to contribute to the consequences of each other; (c) flexibility, i.e., possibility to view the same predicate under both open and closed world interpretations; (d) decidability, preferably of low worst-case complexity, to be used in, e.g., Semantic Web applications.

The Semantic Web Rule Language (SWRL) [41] extends the OWL, providing logic-based rules, which together with stored facts are executed as inputs by the rule engine, which infers new facts as an output. In addition, the rule engine infers new knowledge using forward chaining, which can be used for further inference. The SQWRL (Semantic Query-Enhanced Web Rule Language) [42] is a SWRL-based language for querying OWL ontologies. SQWRL queries operate on known individuals in the currently loaded OWL ontology. The SQWRL provides no way of accessing the information it accumulates from within a rule so query or/and counting or/and computation results cannot be written back to the ontology. There is no way, for example, to insert the result of a recomputed age of some person (based on known birthday of the person and current data) back into the ontology, or to update, e.g., the value of the property *hasPublicationsAmount* for some researcher in the ontology when information about new publications arrives. Such a mechanism could invalidate OWL's OWA and lead to nonmonotonicity [42].

We should admit, however, that appropriate knowledge formalism for the self-managed systems must combine OWA (for cognition and operating with the external environment knowledge) with CWA (for self-awareness and operating with the internal environment knowledge). Therefore OWA is right when talking about knowledge of the world but inappropriate when talking about, e.g., knowledge of knowledge of the world.

6. *Demons* (Procedural Attachments) in Semantic Web

The needed compromise between OWA and CWA, which is how to make the results computed or inferred based on procedural knowledge explicit as declarative knowledge within ontology, is closely related to a compromise between (or a hybrid of) declarative and procedural knowledge. Marvin Minsky in [43] suggested using so called “demons” within *frame* models already in 1974. A frame represents an object or a concept. Attached to the frame is a collection of attributes (slots), filled with values, which can be altered to make the frame correspond to the particular situation. When other frames are used to fill the slots, then as a result we will have a semantic graph of domain objects or/and concepts similarly to what we can get today with RDF and OWL. According to Minsky, frames may also contain procedures, called *demons*, which are activated under prescribed circumstances. Demons are supposed to be attached to some slots in a frame to cause execution of some procedure when accessed. Demons are triggered when, e.g.: (a) a new value is put into a slot; (b) a value is removed from a slot; (c) a slot value is replaced; (d) there is no value present in an instance frame and a value must be computed from a generic frame; and etc.

On the other hand, soon after frames in 1977, Roger Schank [44] proposes *scripts* as a method of representing procedural knowledge based on conceptual primitives (basic actions that people and objects can perform) and their interrelations. Scripts are very much like frames, except the values that fill the slots must be ordered. Frames and script can be easily “married” just because the basic idea involved in both representations is that our knowledge of concepts, events, and situations is organized around *expectations* of their key features. For example one can easily imagine a frame slot (demon in Minsky’s interpretation) with attached script (in Schank’s interpretation) prescribing on how to anytime get/update the actual value of the slot. The Knowledge Representation Language (KRL) [45] has been proposed by Daniel Bobrow and Terry Winograd in 1977 as an attempt to integrate procedural knowledge with a broad base of declarative forms. Procedures in KRL were associated directly with the internal structure of a conceptual object and such procedural attachment assumes a particular operation to be determined by characteristics of the specific entities involved into it.

The frame-based knowledge representation made sufficient influence to the Semantic Web standards (RDF, OWL, etc.); see, e.g., [46] where the frame-based representation has been proposed as a suitable paradigm for building ontologies as well as for the RDF-formalism with its object/attribute/value triples. Since that, however, demons (as one of the most attractive features of the frame model) have never been supported by the RDF data model. Although the simple class and property model of RDF Schema owes much to object-oriented approaches in software engineering, it remains however a purely declarative representation (unlike object-oriented models with member functions on objects, or frames with demons attached to slots) therefore approach taken by RDF has major difference with that taken by traditional object-oriented languages such as C++ or Java [47].

There were some efforts to enable procedural attachments for, e.g., specific calculations or evaluations on the data within ontological knowledge models. For example the FLUMAGIS (www.flumagis.de) project resulted to a prototypic expert system which can be used to provide decision and planning support in the water domain, and its software has been designed as a knowledge-based system. One the major research questions within the project was: how to combine static ontologies with algebraic functions, evaluations, calculations, complex processes leading to simulations and prognosis [48]. The project contributed to creation of a set of plug-ins to the Protégé ontology development environment (protege.stanford.edu) to enable Java procedural attachments through Protégé interface to the ontology. These include: (a) function slots**,** which reckon up certain instance slot values as input, represent the result as their own value, and will be updated automatically if one of the input values has been changed; (b) action buttons to start any procedure, which uses any instance slot values as input, and which also can change any instance slot values as output; (c) constraints (as functions) on slot values. The attachment of a Java class to a slot means that an instance of the Java class is created to be connected with the slot, i.e., there is one Java object for each ontology class, which uses this slot as a template. The Java slots (aka active properties) can be modified at the class level by facets [48].

The need for ontologies capable to represent methods has been also discussed in [49]. It was argued that software agents to be capable to perform their tasks autonomously need methods besides classes and attributes to be also represented in ontologies. The minimum requirements are that the method name is represented along with the types of its arguments and the type of the return value.

7. Other Way around (Semantic Attachments to Procedures): Semantic Web Services

More successful has been an effort to integrate declarative and procedural knowledge by adding declarative semantics to process descriptions, which can be explained by growing popularity of the Web service economy. Web service is a self-contained modular business application that have open, internet-oriented standardized interface. Appropriate Web service standards include SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language), UDDI (Universal Description, Discovery and Integration), WS-BPEL (Web Service Business Process Execution Language), etc. There were several attempts to extend the Web service concept towards a Semantic Web Service, which is a self-descriptive, semantically marked-up software resource that can be published, discovered, composed and executed across the Web in a task-driven way [50], or even to make it proactive (agent-driven) and capable to behave autonomously to increase its utility and to be the subject of negotiation and trade [51].

Among several frameworks to enable Semantic Web Services consider the following ones: OWL-S, WSMO, METEOR-S (WSDL-S, SA-WSDL, SA-REST) and SSWAP.

OWL-S (http://www.daml.org/services/owl-s/) is the first OWL-based Web service ontology, which supplies Web service providers with a core set of constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. Ontology consist of three main parts: the service *profile* for advertising and discovering services; the *process* *model*, which gives a detailed description of a service's operation; and the *grounding*, which provides details on how to interoperate with a service, via messages. The Web Service specifications based on OWL-S are believed to enable the development of software programs that can interpret descriptions of unfamiliar Web Services and then employ those services to satisfy user goals. Therefore the OWL-S markup of Web services is expected to facilitate the automation of Web service tasks, including automated Web service discovery, execution, composition and interoperation [52].

WSMO (http://www.w3.org/Submission/WSMO/) provides a more complete conceptual model comparably to OWL-S as it also addresses aspects such as goals (objectives that a user might have when consulting a service) and mediators or necessary mappings aimed to resolve interoperability problems, incompatibilities and mismatches at different (data, protocol and process) levels [53]. WSMO differentiates between the ontological descriptions (declarative knowledge) of the services and technologies for their execution (procedural knowledge). The service interfaces in WSMO are designed in a way that is suitable for software agents to determine the behavior of the Web service and reason about it [54].

The METEOR-S project (lsdis.cs.uga.edu/projects/meteor-s/) at the LSDIS Lab, University of Georgia, aimed to extend popular within industry Web-services standards (WSDL, UDDI, BPEL4WS) with the Semantic Web technology [55]. As a result, the WSDL-S (semantically enhanced WSDL) has been elaborated as a lightweight approach for adding semantics to Web services. The approach allows integration of semantic and non-semantic descriptions of Web services, assuming that the users must specify translation to OWL of possible special types. For software developers the possibility was given to semantically annotate source files (by exploiting the meta-tags) to enable complete WSDL-S specifications to be generated and automatically published in an enhanced UDDI registry. Special support is provided for automatic generation of OWL-S files from WSDL-S files for grounding, profile and service [56]. Later in 2007, the Semantic Annotations for WSDL and XML Schema (SA-WSDL) (www.w3.org/TR/sawsdl/) mechanism has been elaborated by which concepts from the semantic models that are defined either within or outside the WSDL document can be referenced from within WSDL components as annotations helping to disambiguate the description of Web services during their automatic discovery and composition [57]. Later in 2010, the Semantic Annotations for REST (SA-REST) (www.w3.org/ Submission/SA-REST/) has been elaborated as a microformat to enable ontological metadata to be embedded into HTML/XHTML documents. It is human-friendly as it designed for the humans (developers and annotators) first and for the machines later. SA-REST has two types of properties (block and element property), which meant to distinguish the capability of a property to nest other properties. Attaching explicit meta-data to the API descriptions using SA-REST can significantly improve their faceted search. It also facilitates a user-driven light weight service composition (e.g., mash-ups) [58].

As a sample of more recent activity let us mention the SSWAP (http://sswap.info/), which is OWL ontology specifically designed to describe web services. Services are identified by URIs by mapping from their inputs to outputs, and the SSWAP ontology defines a set of terms to describe this transformation. The same representation is also used for service search and service execution requests and responses. A service execution request fills in the value of an input parameter and leaves the output value blank to be filled as a result of the service execution. Thus, SSWAP defines a protocol where clients and servers exchange OWL documents that contain needed RDF graph structure for accomplishing the tasks of services discovery and execution [59].

Summing it up we may admit that these approaches target mostly semantic support and automation of a customer (directly or through some application) – a service provider relationships and are not so suitable for the self-management systems. According to our previous consideration, a self-managed system should, if appropriate, be capable to discover and utilize external web services for the self-management needs (e.g., for self-reconfiguration) or should also be aware and when needed utilize own services (self-services) for the same purpose. The former requires a capability for an external service to securely access the configuration of the system and make necessary changes within it, which definitely may be considered as a new requirement to the (semantic) Web service community. The latter will require appropriate ontology and annotations of the internal system capabilities as services and can be in principle supported by one of the Semantic Web services approaches described above. However if we consider a knowledge-based system in general and its knowledge in particular as a subject for self-reconfiguration, then this will be challenging from the both aspects if to apply traditional technologies due to lack of appropriate services to manipulate with knowledge in fully automated manner.

8. Introducing Executable Knowledge and Executable Knowledge Computing

To address the challenges discussed earlier in the paper we introduce the concept of executable knowledge for autonomic or self-management systems development according to Semantic Web standards. The used knowledge representation model based on directed labeled RDF-graphs on top of a set of triples “subject – predicate – object” (or “resource – property – value”) considers nodes of the graphs as blank nodes, literals or URI (Uniform Resource Identifier) references. The traditional role of the URI reference has always been an identifier pointing at the name and location of a web resource. A new role defined lately by the W3C is to identify the content located at the referenced Web-resource if the resource can be dereferenced (http://www.w3.org/2001/tag/doc/httpRange-14/HttpRange-14.html) do not provide capacities for ontology evolution.

We extend the semantics of the RDF triple more by defining the third possible role for URI: to be a procedure identifier. This adds a procedural component to the RDF-based data model giving a possibility to define procedures for automatic ontology evolution and instantiation. We called knowledge utilizing the proposed data model *executable* with respect to the manner a reasoner works with it (executing accommodated instructions).

The new model is an ontology which introduces a new property type named “executable property” (in addition to the object and datatype properties), which value is not an explicitly defined data but an instruction on how to get it. In this case the executable triple components get roles of “resource – property – procedure (query)”. According to the basic RDF data model a procedure is a resource with several properties for storing instructions and a cash of the computed values. As it is shown in Figure 1 Resource_A has a property and, in order to get its value, instructions, which are located in the Resource_D, should be executed by a reasoner supporting inference over executable knowledge.

Two immediate advantages of the extension one may expect are: (a) a triple will always implicitly keep knowledge about most recent value for the property because query to some data storage or to some computational function will be executed only on demand when needed and the latest information will be delivered; (b) a query may be written according to different standards, data representation types, models and schemas so that heterogeneity of original sources of data and capabilities will not be a problem.

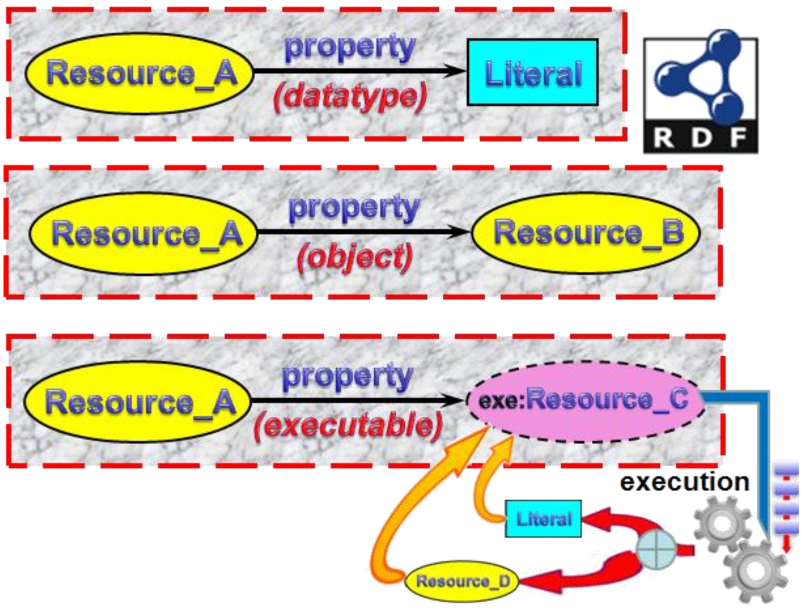


Figure 1. Extending the traditional RDF property types (datatype or object property) by a new one "executable property", which value (resource or literal) is computed on access following the instructions (Resource_C in the picture, which prefix *exe* indicates an executable resource).

In [60] we have already demonstrated how a similar approach can be used for creating semantic mash-ups of the reality data and the business intelligence functions (we called the concept as the "executable reality"). There we based the implementation on the S-APL (Semantic Agent Programming Language) [61]. In this paper we try to discuss the implementation issues following the traditional semantic technology standards (RDF, OWL).

The concept of executable knowledge can be considered as such kind of hybrid of declarative and procedural knowledge, where "executing" knowledge, one actually transforms tacit (procedural) knowledge into explicit (declarative one). Therefore an executable knowledge contains explicit procedural (meta-) knowledge on how to acquire (or compute) declarative knowledge. Such capability means that the executable knowledge is naturally self-configurable knowledge (or more generally – self-managed knowledge). Therefore, taking into account that the procedural attachments can be treated if needed as self-management instructions (for example, self-evaluation, self-configuration, verification, filtering, decontextualizing, reasoning, computing, merging, compressing, reporting, visualizing, etc.), we may consider the executable knowledge as a self-management enabled one. Let us call the process of executable knowledge (self-) management as *Executable Knowledge Computing* (Figure2).

Assume some knowledge storage, which in addition to some domain knowledge also has an explicit knowledge management instruction in the form of executable knowledge. If the storage is equipped with a "knowledge processor" (i.e., an engine or an agent capability to execute the instruction scripts), then the knowledge will behave as a self-management entity (Figure 2).

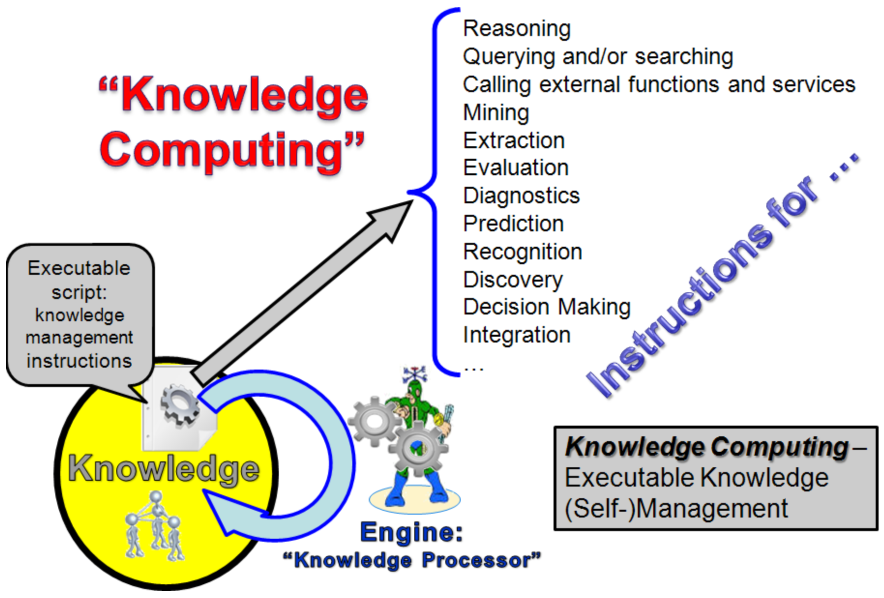

"Knowledge Computing"
Reasoning
Querying and/or searching
Calling external functions and services
Mining
Extraction
Evaluation
Diagnostics
Prediction
Recognition
Discovery
Decision Making
Integration
…
Instructions for …
Executable script: knowledge management instructions
Knowledge
Engine: "Knowledge Processor"
Knowledge Computing – Executable Knowledge (Self-)Management

Figure 2. The concept of "Executable Knowledge Computing" illustrated.

It is important to mention that executable scripts may refer also to external service/capability providers, therefore some executions may be exported and performed remotely and the results will be consumed on return through the knowledge processor back to knowledge.

The approach of executable knowledge computing allows addressing a number of problems considered as a bottleneck for the Linking Open Data:

- the higher index of links integrity (defined as a qualitative property that is given when all links within and between a set of data sources are valid and deliver the result data intended by the link creator [33]) is reached due to reducing the number of the broken links possible types. A link will always have a valid target computed on fly which excludes the crucial influence of removing and moving previously defined link targets.
- Instance-level coreference (discussed in [62]) is enabled by executable knowledge computing by inferring schema-level and instance-level relations between concepts and properties of two different repositories providing data-level and schema-level evidence.
- Automatic ontology evolution and instantiation enabled by the new data model reduces the possibility of ontologies overloading with lots of concepts and instances or indexed Web documents and is a possible solution for rebalancing graph structures [31].

9. Pilot Implementation and the Example

We define the concept of executable property as an enabling technology to interconnect ontology-based (semantic) portals with credible data sources and services in the most suitable and flexible way to update knowledge bases of the portals automatically in real time. At this stage of our research we have considered 4 basic types of executable properties (see Figure 3), which are capable to update their values by:

1) getting data from traditional databases performing an SQL query;
2) making simple mathematical computations over other known values from local semantic storage;
3) getting data as a result of SPARQL queries;
4) getting data as query results from external Web services.

Executable knowledge model is an ontology defining new executable classes and properties. To process it an executable knowledge specific processor is needed. We developed a plug-in for Protégé ontology development environment (protege.stanford.edu), which supports the new property type and provides reasoning tools over the executable part of the ontology.

Depending on the subtype of the executable property a certain procedure for executing and resolving should be applied.

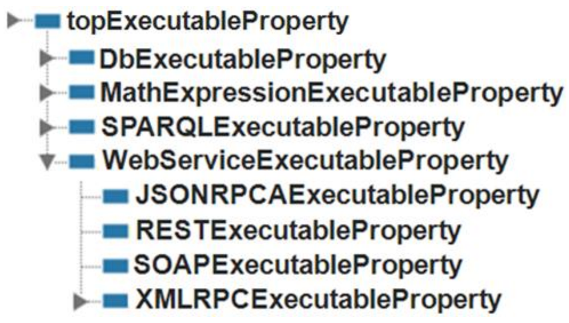


Figure 3. A hierarchy of the Executable Property types.

Each of the subproperties needs a unique mechanism for computing the value and requires different parameters:

1) Top Executable Property is an abstract super property for all possible executable properties. It contains a description of the common computing mechanism, which will be inherited by all subproperties. For correct specification of the input parameters the reasoner gets access to a particular instance of the class of the computed executable property. Each inheritor of the Top Executable Property has its own syntax for formal description of the needed parameters' values. Generalization of the executable property domain is implemented by use of operator "this" defined in the Top Executable Property. For instance, reference "this.name" points out to the value of the property "name" of the currently processed instance. Such notation allows implementing a specific syntax of the query within the properties of the inheritors. Executable properties may have a set of possible ranges, and for each of them parameters should be defined separately.

2) "DataBase Executable Property" allows getting data from a database in response to an SQL query. Parameters for such property are: database server, access parameters (server address, login and password), and the SQL query itself.

3) "SPARQL Executable Property" enables to describe a SPARQL query as a procedure of the property value computing.

4) "Mathematical Executable Property" enables computing a relatively simple mathematical formula (function). The function itself is specified as a parameter for the executable property. Such property can operate with constants as well as with the values of the various resource properties. The function can have variables referring to other executable properties.

5) "Web Service Executable Property" enables querying a remote Web service. Such query will require the following parameters: (a) access data (address, login, password, etc.); (b) identifier for the function called; (c) identifiers and values for the parameters needed for the function (values of other properties of the original resource can be taken as the parameters). Various possible protocols for the Web-services are handled by creating subclasses of the *Web Service Executable Property* separately for each protocol.

In spite of the expectations that the executable property concept may expand the applicability of the ontology-based systems and improve their interoperability, the approach has also some weaknesses related to the reduced ontology performance because computing the values for the executable property on-the-fly may create some delays in getting output data. Cashing mechanisms are needed to handle history of queried executable property values to fasten in some cases calculations in real time.

Let us consider calculation of a scientist`s rank as an example of an executable property management. Assume that three different citation indexes (h-index, g-index and k-index) of some scientist are taken from different sources and the final value "ScientisRank" will be computed as a mean value of those. Class Scientist in the ontology will look as shown in Figure 4.

Imagine that the h-index has to be calculated as a result of SPARQL query; the g-index is computed as the result of SQL query; the k-index is obtained due to querying web-service (XML RPC); and finally the ScientistRank is computed as a mean value of the three above numbers.

Taking into account that query transfer protocols are different for different Web services we have developed supporting tools for 4 types of interaction with Web services: “JSON RPC”, “REST”, “SOAP” and “XML RPC”. All of them have a similar set of parameters for querying a service but interact with the service based on different protocols. In our example we will show in detail the option of a query transfer according to the “XML RPC” standard.

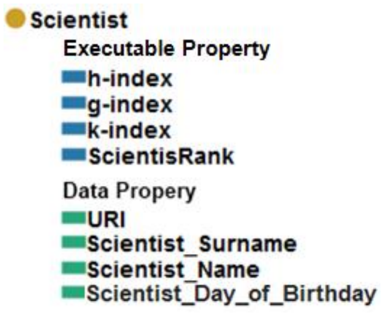
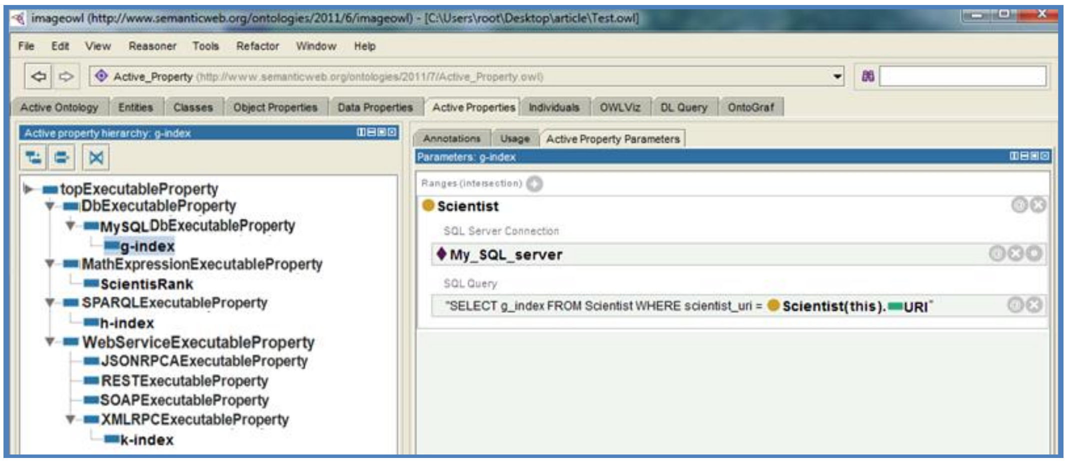

Scientist
Executable Property
h-index
g-index
k-index
ScientisRank
Data Propery
URI
Scientist_Surname
Scientist_Name
Scientist_Day_of_Birthday

Figure 4. Properties of the “Scientist” class in the example.

For getting the g-index the “DataBase Executable Property” is used. Parameters of this property are shown in the Protégé screenshot in Figure 5. One needs to specify two parameters to get the value of “MySQLDbActive Property”. The first one defines connection with the database. For that a special object of the SQL_Server_Connection class is created with the properties shown in Figure 6. Such object contains all necessary information for connecting with the database server. Connection itself can be performed in advance for fast support of possible set of queries. The second parameter (the SQL query itself) can be entered in a special window of the editor, which allows combining textual descriptions with the references to classes and properties.


imageowl (http://www.semanticweb.org/ontologies/2011/6/imageowl) - [C:\Users\root\Desktop\article\Test.owl]
File Edit View Reasoner Tools Refactor Window Help
Active_Property (http://www.semanticweb.org/ontologies/2011/7/Active_Property.owl)
Active Ontology Entities Classes Object Properties Data Properties Active Properties Individuals OWLViz DL Query OntoGraf
Active property hierarchy: g-index
Annotations Usage Active Property Parameters
Parameters: g-index
Ranges (intersection)

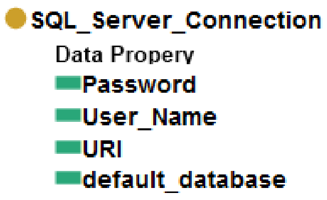
topExecutableProperty
- DbExecutableProperty
  - MySQLDbExecutableProperty
    - g-index
- MathExpressionExecutableProperty
  - ScientisRank
- SPARQLExecutableProperty
  - h-index
- WebServiceExecutableProperty
  - JSONRPCAExecutableProperty
  - RESTExecutableProperty
  - SOAPExecutableProperty
  - XMLRPCExecutableProperty
    - k-index

Scientist
SQL Server Connection
My_SQL_server
SQL Query
"SELECT g_index FROM Scientist WHERE scientist_uri = Scientist(this).URI"

Figure 5. Protégé screenshot for the g-index (executable property) in the example.

For each particular "Domain" specification for an executable property there is a possibility to define needed set of parameters. If for some "Domain" such parameters will be missing, then, if queried, such property will return an empty line as a value.

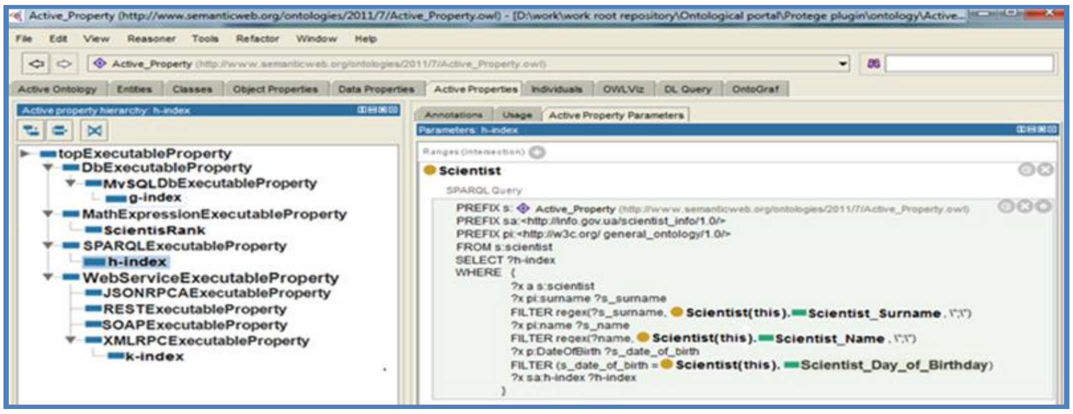
Such type of executable property has just one parameter (textual field), which may refer to the appropriate ontology, to the pointer “this” and to the properties of the appropriate instance of the class. Such textual description has to be processed with the modified SPARQL parser and appropriate query will be resolved.



● SQL_Server_Connection
Data Propery
🟩 Password
🟩 User_Name
🟩 URI
🟩 default_database

Figure 6. Properties of the SQL_Server_Connection class in the example.

For obtaining the h-index we use “SPARQL Executable Property”, parameters of which are shown in the screenshot in the Figure 7.

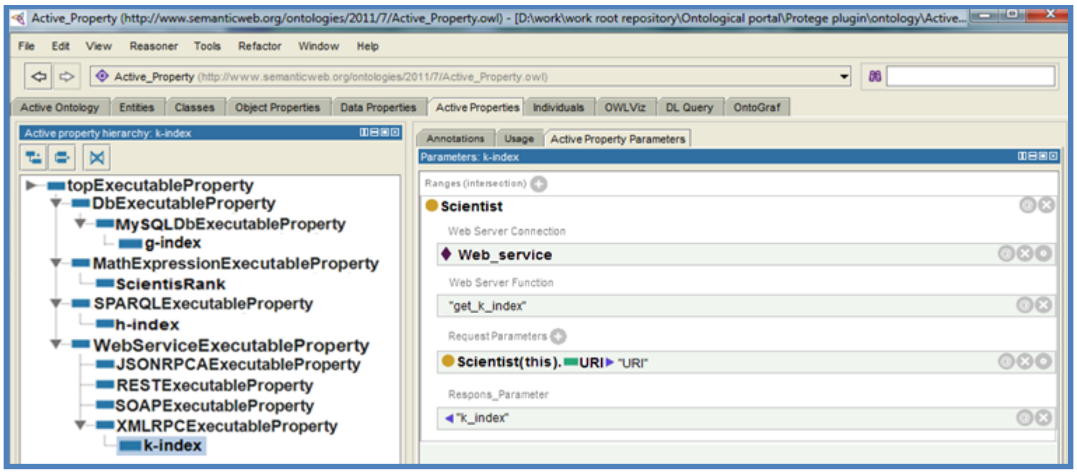

Active_Property (http://www.semanticweb.org/ontologies/2011/7/Active_Property.owl) - [D:\work\work root repository\Ontological portal\Protege plugin\ontology\Active...
File Edit View Reasoner Tools Refactor Window Help
Active_Property (http://www.semanticweb.org/ontologies/2011/7/Active_Property.owl)
Active Ontology Entities Classes Object Properties Data Properties Active Properties Individuals OWLViz DL Query OntoGraf
Active property hierarchy: h-index
topExecutableProperty
DbExecutableProperty
MySQLDbExecutableProperty
g-index
MathExpressionExecutableProperty
ScientisRank
SPARQLExecutableProperty
h-index
WebServiceExecutableProperty
JSONRPCAExecutableProperty
RESTExecutableProperty
SOAPExecutableProperty
XMLRPCExecutableProperty
k-index
Annotations Usage Active Property Parameters
Parameters: h-index
Ranges (intersection)
Scientist
SPARQL Query

```
PREFIX s: Active_Property (http://www.semanticweb.org/ontologies/2011/7/Active_Property.owl)
PREFIX sa:<http://info.gov.ua/scientist_info/1.0/>
PREFIX pi:<http://w3c.org/ general_ontology/1.0/>
FROM s:scientist
SELECT ?h-index
WHERE  {
        ?x a s:scientist
        ?x pi:surname ?s_surname
        FILTER regex(?s_surname, Scientist(this).Scientist_Surname , \"",\")
        ?x pi:name ?s_name
        FILTER regex(?name, Scientist(this).Scientist_Name , \"",\")
        ?x p:DateOfBirth ?s_date_of_birth
        FILTER (s_date_of_birth = Scientist(this). Scientist_Day_of_Birthday)
        ?x sa:h-index ?h-index
      }
```

Figure 7. Protégé screenshot for the h-index (executable property) in the example.

For getting the k-index we use the “Web Service Executable Property”. To support most popular types of Web services we have implemented various subproperties for the "Web Service Executable Property" according to the hierarchy from Figure 3. The major difference among the mentioned subproperties is the information transfer protocol between the client and the service. In our example we compute the value for the k-index utilizing Web service according to the "XML RPC" standard. Parameters for the “XML RPC Executable Property” are shown in the screenshot in Figure 8.

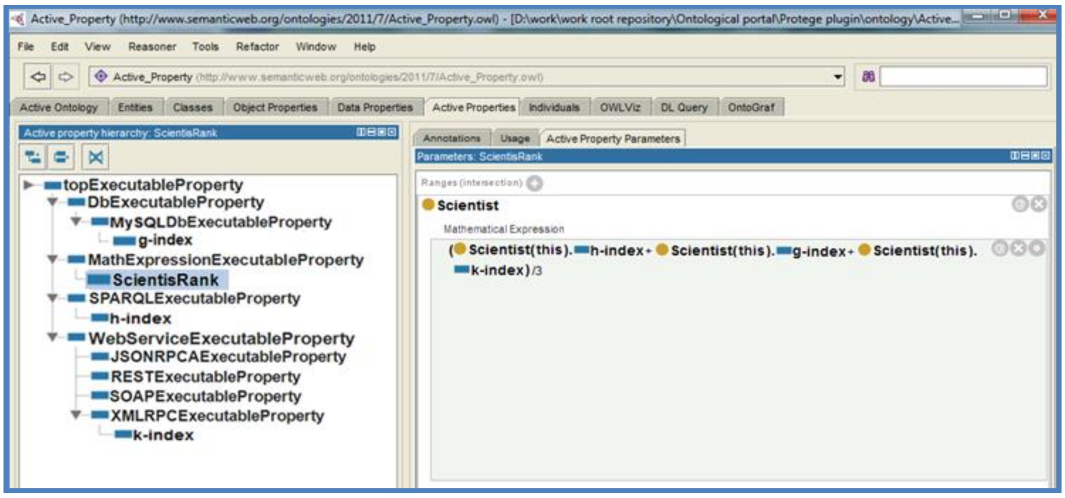
One can see that such type of Web service requires instance of class "Web_Service_Connection" as a parameter, which contains information about the Web service address and access keys to it. Parameter “Web service function” specifies the required function of the Web service. The plug-in allows compiling various sets of parameters needed for querying the required function. The value for each of these parameters can be provided due to special editor through which one specifies the name of needed parameter and its value. The value is a constant as a rule or a reference to some other property value already specified for target instance. For specification of the Web service output parameter we used “Respons_Paramenter” field.


Active_Property (http://www.semanticweb.org/ontologies/2011/7/Active_Property.owl) - [D:\work\work root repository\Ontological portal\Protege plugin\ontology\Active...
File Edit View Reasoner Tools Refactor Window Help
Active_Property (http://www.semanticweb.org/ontologies/2011/7/Active_Property.owl)
Active Ontology Entities Classes Object Properties Data Properties Active Properties Individuals OWLViz DL Query OntoGraf
Active property hierarchy: k-index
Annotations Usage Active Property Parameters
Parameters: k-index
topExecutableProperty
DbExecutableProperty
MySQLDbExecutableProperty
g-index
MathExpressionExecutableProperty
ScientisRank
SPARQLExecutableProperty
h-index
WebServiceExecutableProperty
JSONRPCAExecutableProperty
RESTExecutableProperty
SOAPExecutableProperty
XMLRPCExecutableProperty
k-index

Ranges (intersection)
Scientist
Web Server Connection
Web_service
Web Server Function
"get_k_index"
Request Parameters
Scientist(this).URI "URI"
Respons_Parameter
"k_index"

Figure 8. Protégé screenshot for the k-index (executable property) in the example.

For calculating the integrated value (in our example - average) for the indexes we create the property “ScientisRank”, which is computable. Parameters for such “Mathematical Executable Property” are shown in the screenshot in Figure 9. The only needed parameter for such type of executable property is the mathematical expression, which may contain constants or variables (properties of the target instance).


Active_Property (http://www.semanticweb.org/ontologies/2011/7/Active_Property.owl) - [D:\work\work root repository\Ontological portal\Protege plugin\ontology\Active...
File Edit View Reasoner Tools Refactor Window Help
Active_Property (http://www.semanticweb.org/ontologies/2011/7/Active_Property.owl)
Active Ontology Entities Classes Object Properties Data Properties Active Properties Individuals OWLViz DL Query OntoGraf
Active property hierarchy: ScientisRank
Annotations Usage Active Property Parameters
Parameters: ScientisRank
topExecutableProperty
DbExecutableProperty
MySQLDbExecutableProperty
g-index
MathExpressionExecutableProperty
ScientisRank
SPARQLExecutableProperty
h-index
WebServiceExecutableProperty
JSONRPCAExecutableProperty
RESTExecutableProperty
SOAPExecutableProperty
XMLRPCExecutableProperty
k-index

Ranges (intersection)
Scientist
Mathematical Expression
(Scientist(this).h-index+Scientist(this).g-index+Scientist(this).k-index)/3

Figure 9. Protégé screenshot for the Scientist Rank (executable property) computing in the example.

The major ongoing activity, which is based on the implementation of the Knowledge Computing concept, is the EU Tempus-IV Project TRUST: "Towards Trust in Quality Assurance Systems" (516935-TEMPUS-1-2011) (www.dovira.eu) [62-65]. The overall goal of the TRUST project is to support the reforms of Ukrainian Higher Education (HE) by introducing a comprehensive and transparent Quality Assurance (QA) framework for all HE institutions (HEI) and QA organizations, which is based on the knowledge triangle ("education-research-innovation"). A Knowledge Ecosystem solution autonomously enables, supports and automates the QA activities and transactions between HEIs, different national and international QA actors, students and different stakeholders and supports various forms of information exchange and knowledge sharing. The framework is assumed to guarantee trust between all the QA players and society by ensuring that all QA procedures will be based not only on credible, transparent and relevant sources of information but also on explainable and in the same time executable decision-making techniques documented in a common portal. A trusted QA system should be based as much as possible on external objective evaluations. However because it is difficult to immediately utilize expensive experience of external evaluators in Ukrainian QA system we are making all the academic information around QA processes available and transparent to national and international academic community and combine it with other publicly available, retrievable via Web-services information and automatically computable quality indicators on top of it within the ecosystem.

The objectives above are supported by a self-managed knowledge Portal (www.portal.dovira.eu), named TRUST [63-65, 60], which is a work-in-progress, and which enables, supports and automates the activities, information flows and transactions within the ecosystem of individuals, HEIs, and QA organizations. Provided IT-support of QA enables: machine-processable and executable QA-related information; management of globally distributed and heterogeneous QA-related data collections and Web-services; QA-related automated knowledge transfer through intelligent information retrieval, extraction, sharing, reuse and integration. To achieve this, the knowledge needed for QA is organized according to the Executable Knowledge concept and it is augmented in several dimensions: (1) to allow anybody adding a personal QA technique (a "Quality Calculator") or evaluation criteria (i.e., executable properties as described above) to the knowledge base and to get a personalized view on the quality status (in absolute or relative scales) of any educational organization or any educational outcome. As a result such executable knowledge becomes in a way a “Smart Knowledge” (i.e., enable self-evaluation formulas, QA procedures and techniques to be proactive knowledge instances, to be self-descriptive, extendable, self-managed and reusable); (2) to make the results more trustful such executable knowledge must also be a “Cross-Validated Knowledge” (i.e., providing Service-Oriented Architecture for automatic update of the values of various quality indicators by taking them from external Web-based sources (portals, databases, etc.), such as, e.g., ISI Web of Knowledge, Google Scholar, etc., externalizing and internationalizing various quality monitoring activities); (3) to help a user to see the reasons behind good or bad performance we need our knowledge to be also a “Self-Explanatory Knowledge” (that provides automated support for detailed explanation of every calculated or inferred evaluations); (4) to automate the interpretation of the computed evaluations in different situations we need such executable knowledge to perform also as a “Context-Aware Knowledge” (i.e., capable o utilization of formalized knowledge about context (local, regional, national, international, etc., for providing more grounded evaluations in a particular context).

Executable knowledge was integrated into the knowledge base of the Portal at the stage of its piloting. This increased the speed of the access of the ontology to relevant linked data and its update. The speed of data logging, for instance, was increased in 9,5 times by use of executable SQL-properties which allowed to add data about logging into the base. Piloting of the Portal was done on 50000 instances. The system was implemented on the basis of JBoss (volumes of the information about events correspond to the average value of a JBoss server 7.1).

10. Conclusions

In this paper we have discussed current trends in Information and Communication Technologies figuring out the needs for autonomic and self-managed solutions related to knowledge economy and provided some basic characteristics and requirements to appropriate self-managed systems. These include the need in a hybrid of declarative knowledge and procedural (executable) one based on smart compromise between the Open World and the Closed World assumptions. If to consider a knowledge-based system in general and its knowledge in particular as a subject for self-reconfiguration, then there will be challenging to apply traditional technologies due to lack of appropriate services to manipulate with knowledge in fully automated manner. We introduced the concepts of executable knowledge and knowledge computing on the basis of adding an executable property to the traditionally used (datatype and object) properties within the RDF model. Two immediate advantages of the extension are: (a) an RDF triple will always implicitly keep knowledge about most recent value for the property because query to some data storage or to some computational function will be executed only on demand when needed and the latest information will be delivered; (b) query may be written according to different standards, data representation types, models and schemas so that heterogeneity of original sources of data and capabilities will not be a problem. Therefore knowledge can be managed autonomously and “queried” in real time by the executable RDF links. We also present some pilot (proof-of-concept) implementation of the executable knowledge as a plug-in to Protégé ontology development environment and briefly present more solid implementation work-in-progress activity related to the self-managed national educational resources.

Proposed concepts of the executable properties, executable knowledge and executable knowledge computing allow integrating different technologies within one ontology-driven domain specification. Such approach may transfer any massive RDF storage into a flexible, dynamic, self-managed knowledge base, which will always contain actual facts about the domain objects. The potential of self-organized semantic storage services has been recently discussed in [61] on the basis of thorough literature survey. It was argued that the analyzed approaches and their underlying technologies were unable to distribute large amounts of semantic information in a generic way while still being able to react on changing environmental infrastructure. Therefore self-organization in a distributed knowledge storage system is still an important and challenged issue nowadays.

We believe there might be many other application domains (to be investigated still) where the concept of executable knowledge computing might be useful, e.g.: self-managed web sites/pages; self-managed content for e-learning and knowledge transfer; Data Journalism (database journalism and data-driven journalism); Cloud Computing (knowledge-as-a-service), Linked Data and Business Intelligence on top of it, and many others.

References

[1] D. Lyon, *The Information Society: Issues and Illusions*, Blackwell Publishers Inc., Cambridge, MA, USA, 1988, 196 pp.

[2] M. Castells, *The Rise of the Network Society*, Wiley-Blackwell, Second Edition, 2010.

[3] A. Simard, Knowledge Markets: More than Providers and Users, *Transactions on Advanced Research*, Vol. 2, No. 2, July 2006, IPSI BgD, pp. 4-9.

[4] J. Kelly, From Knowledge Management to Knowledge Ecosystem, In: *Learn to Adapt Links*, 17.10.2008, Available in: http://learn2adapt.com/blog/2008/10/17/from-knowledge-management-to-knowledge-ecosystem.

[5] H. Jenkins, *Convergence Culture: Where Old and New Media Collide*, NY Univ. Press, 2006, 308 pp.

[6] W.W. Powell, and K. Snellman, The Knowledge Economy, *Annual Review of Sociology,* Vol. 30, 2004, pp. 199-220.

[7] T.-B. Lee, *Linked Data*, 27.07.2006, Available in: http://www.w3.org/DesignIssues/LinkedData.html.

[8] T. Heath and C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*, Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool, 2011, 136 pp.

[9] R. Cyganiak, and A. Jentzsch, *Linking Open Data Cloud Diagram*, September 2011, available in: http://lod-cloud.net/.

[10] A. Ankolekar, M. Krotzsch, T. Tran, and D. Vrandecic, The Two Cultures: Mashing Up Web 2.0 and the Semantic Web, In: *Proceedings of the 16th International Conference on World Wide Web* (WWW '07), ACM, NY, USA, 2007, 825-834.

[11] M. Zollo, and S.G. Winter, Deliberate Learning and the Evolution of Dynamic Capabilities, *Organization Science*, Vol. 13 No. 3, May/June 2002, pp. 339-351.

[12] *The Impact of the Crisis on ICT and ICT-Related Employment,* In: OECD (Organization for Economic Cooperation and Development) Report, October 2009, available in: http://www.oecd.org/internet/interneteconomy/43969700.pdf .

[13] W. Fan, A. Bifet, Q. Yang, P. Yu, Foreword, In: *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Beijing, China, August 12-16, 2012.

[14] L. Heuser, Vision of a Web-Based Service Society, *SAP Research Web pages*, October 8, 2008, available in: http://en.sap.info/vision-of-a-web-based-service-society/11099.

[15] V. Terziyan, Bridging Webs for Future Business: "Everything-as-a-User", In: *Presentation at Open Discussion: "Business through Technologies or Technologies through Business" during BUSTECH-2011*, Rome, Italy, 29 September, available in http://www.iaria.org/conferences2011/filesBUSTECH11/Terziyan_Rome_2011.pdf.

[16] E. Ormala (Nokia Vice-President), Innovation Calls for People, In: *Presentation "Changing Innovation Landscape: The Role of Business and Universities*", Jyvaskyla, Finland, 20.03.2012.

[17] D. Weinberger, *Too Big to Know. Rethinking Knowledge Now That the Facts Aren't the Facts, Experts Are Everywhere, and the Smartest Person in the Room Is the Room*, Basic Books Publisher, 1st Edition, January 2012, 231 pp.

[18] V. Maracine, and E. Scarlat, Dynamic Knowledge and Healthcare Knowledge Ecosystems, In: *Electronic Journal of Knowledge Management*, Vol. 7, No. 1, 2009, pp. 99-110.

[19] Autonomic Computing: An Overview: The 8 Elements, In: IBM Perspective on the State of Information Technology, *Web Site of IBM Research*, Accessed 28 September, 2012, Available in: http://www.research.ibm.com/autonomic/overview/elements.html .

[20] M., Kim, J., Jeong, S., Park, From Product Lines to Self-Managed Systems: an Architecture-Based Runtime Reconfiguration Framework, In: *Proceedings of the Workshop on Design and Evolution of Autonomic Application Software (DEAS '05)*, NY, USA, ACM SIGSOFT Software Engineering Notes, Vol. 30, No. 4., July 2005, pp. 1-7.

[21] J., Kramer, J., Magee: Self-Managed Systems: an Architectural Challenge, In: L.C. Briand, A.L. Wolf (Eds.): *International Conference on Software Engineering, ISCE 2007, Workshop on the Future of Software Engineering*, May 23-25, 2007, MN, USA, pp. 259-268.

[22] V. Terziyan, Semantic Web Services for Smart Devices Based on Mobile Agents, In: *International Journal of Intelligent Information Technologies*, Vol. 1, No. 2, April-June 2005, Idea Group, pp. 43-55.

[23] V. Terziyan, and S. Puuronen, Knowledge Acquisition Based on Semantic Balance of Internal and External Knowledge, In: I. Imam, Y.Kondratoff, A. El-Dessouki and A. Moonis (Eds.), Multiple Approaches to Intelligent Systems, *Lecture Notes in Artificial Intelligence,* Springer-Verlag, V. 1611, 1999, pp. 353-361.

[24] *G.V. Wyk, A Postmodern Metatheory of Knowledge As a System, Trafford Publ., 2004, 302 pp.*

[25] R.J. Hollingdale, *Western Philosophy: An Introduction*, Kahn & Averill, 1979, 166 pp.

[26] G. Bateson, *Steps to an Ecology of Mind*, C Jason Aronson Inc., Northvale, New Jersey, London, 1972, 361 pp.

[27] M. Randles, A. Taleb-Bendiab, P. Miseldine and A. Laws, Adjustable Deliberation of Self-Managing Systems, In: *Proceedings of the 12th IEEE International Conference on the Engineering of Computer-Based Systems,* April 2005, pp. 449 – 456.

[28] D. Green, Knowledge Management for a Postmodern Workforce: Rethinking Leadership Styles in the Public Sector, *Journal of Strategic Leadership*, Vol. 1, No. 1, 2008, pp. 16-24.

[29] T. Cofino, Y. Doganata, Y. Drissi, T. Fin, L. Kozakov, and M. Laker, Towards Knowledge Management In Autonomic Systems, In: *Proceedings of the Eighth IEEE International Symposium on Computers and Communication*, June-July 2003, pp. 789 - 794.

[30] Alferes, J., Pereira, L., Przymusinska, H., Przymusinski, T., Quaresma, P., Dynamic Knowledge Representation and its Applications, In: *Proceedings of the 9th International Conference on Artificial Intelligence: Methodology, Systems, and Applications (AIMSA '00)*, Varna, Bulgaria, September 20-23, 2000, LNCS, Vol. 1904, Springer, pp. 1-10.

[31] Brandao, S., Oliveira, J., & de Souza, J. M. Web Knowledge Representation with Autonomic Ontologies.

[32] Brandao, S. N., Rodrigues, S. A., Silva, T., Araujo, L., & de Souza, J. M. (2013, March). Autonomic Ontologies for Governamental Knowledge Base. In ICAS 2013, The Ninth International Conference on Autonomic and Autonomous Systems (pp. 96-102).

[33] Haslhofer, B., & Popitsch, N. (2009, August). DSNotify-detecting and fixing broken links in linked data sets. In Database and Expert Systems Application, 2009. DEXA'09. 20th International Workshop on (pp. 89-93). IEEE.

[34] Umbrich, J., Hausenblas, M., Hogan, A., Polleres, A., & Decker, S. (2010). Towards dataset dynamics: Change frequency of linked open data sources.

[35] Passant, A., & Mendes, P. N. (2010, May). sparqlPuSH: Proactive Notification of Data Updates in RDF Stores Using PubSubHubbub. In SFSW.

[36] Alferes, J., Pereira, L., Przymusinska, H., Przymusinski, T., LUPS - A Language for Updating Logic Programs, In: *Proceedings of the LPNMR'99*, El Paso, Texas USA, December 2-4, 1999, LNAI, Vol. 1730, Springer, pp. 162-176.

[37] Berge, T., Hezewijk, R., Procedural and Declarative Knowledge. An Evolutionary Perspective, Theory & Psychology, 1999, *Sage Publications*, Vol. 9, No. 5, pp. 605–624.

[38] M. Knorr, J.J. Alferes, and P. Hitzler, Local Closed World Reasoning with Description Logics under the Well-Founded Semantics, *Artificial Intelligence*, Vol. 175, Ns. 9-10, 2011, pp. 1528-1554.

[39] I. Horrocks and P. F. Patel-Schneider. A proposal for an OWL rules language. In: *Proceedings of the 13th Int. World Wide Web Conference (WWW 2004)*, ACM, 2004, pp. 723-731.

[40] B. Motik and R. Rosati, Reconciling Description Logics and Rules. *Journal of the ACM*, Vol. 57, No. 5, 2010.

[41] Web Rule Language: Combining OWL and RuleML, *W3C Member Submission*, 21 May 2004, Available in: http://www.w3.org/Submission/SWRL/.

[42] M.J. O'Connor, and A. Das, SQWRL: a Query Language for OWL, In: *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED-2009)*, Chantilly, VA, USA, 2009.

[43] Minsky, M., *A Framework for Representing Knowledge*, In: P. Winston (ed.), The Psychology of Comp. Vision, McGraw-Hill, 1975.

[44] R.C. Schank, and R.P. Abelson, *Scripts, Plans, Goals, and Understanding.* Hillsdale, N.J.: Lawrence Erlbaum Associates, 1977.

[45] D.G. Bobrow, and T. Winograd, An Overview of KRL, a Knowledge Representation Language, *Cognitive Science*, Vol. 1, No. 1, 1977, pp. 3-46.

[46] O. Lassila, and D. McGuinness, The Role of Frame-Based Representation on the Semantic Web, *Knowledge Systems Laboratory Report KSL-01-02*, Stanford University, 2001.

[47] N. Gibbins, and N. Shadbolt, Resource Description Framework (RDF), In M.J.Bates (ed.): *Understanding Information Retrieval Systems Management, Types and Standards*, Auerbach Publications, 2011, pp. 659-670.

[48] R. Borchert, How Can a Knowledge Base Run Executables on the Frame Level? In: *Proceedings of the Sixth International Protégé Workshop*, 7-9 July, 2003, Manchester, England.

[49] L. Mota, L. Botelho, H. Mendes, and A. Lopes, O3F: an Object Oriented Ontology Framework, In: *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2003)*, Melbourne, Australia, 14-18 July, 2003, pp. 639-646.

[50] S. Arroyo, R. Lara, J. Gomez, D. Berka, Y. Ding, and D. Fensel, *Semantic Aspects of Web Services: Practical Handbook of Internet Computing*, Chapman & Hall and CRC Press, 2004.

[51] V. Ermolayev, N. Keberle, S. Plaksin, O. Kononenko, and V. Terziyan, Towards a Framework for Agent-Enabled Semantic Web Service Composition, *International Journal of Web Services Research*, Vol. 1, No. 3 , 2004, pp. 63-87.

[52] D, Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D.L. McGuinness, E. Sirin, N. Srinivasan, Bringing Semantics to Web Services with OWL-S, *World Wide Web*, Vol. 10, No. 3, 2007, Springer, pp. 243-277.

[53] R. Lara, D. Roman, A. Polleres, and D. Fensel, A Conceptual Comparison of WSMO and OWL-S Web Services, In: *Proceedings of the ECOWS-2004*, Erfurt, Germany, September 27-30, 2004, Lecture Notes in Computer Science, Vol. 3250, Springer, pp. 254 -269.

[54] R. Dumitru, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel, Web Service Modeling Ontology, *Applied Ontology*, Vol. 1, No. 1, 2005, IOS Press, 77–106.

[55] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller, Adding Semantics to Web Services Standards, In: *Proceedings of the International Conference on Web Services (ICWS-2003)*, Las Vegas, USA, June 23 - 26, 2003, pp.395-401.

[56] R. Akkiraju, J. Farrell, J.Miller, M. Nagarajan, M. Schmidt, A. Sheth, and K. Verma, Web Service Semantics - WSDL-S, In: *The Joint UGA-IBM Technical Note*, Ver. 1.0, April 18, 2005, available in: http://lsdis.cs.uga.edu/projects/METEOR-S/WSDL-S.

[57] K. Verma, and A. Sheth, Semantically Annotating a Web Service, *IEEE Internet Computing*, Vol. 11, No. 2, March-April 2007, pp. 83-85.

[58] Sheth, K. Gomadam, J. Lathem, SA-REST: Semantically Interoperable and Easier-to-Use Services and Mashups, *Internet Computing*, **Vol.** 11 No. 6, IEEE, 2007, **pp.** 91-94.

[59] D. Gessler, G. Schiltz, G. May, S. Avraham, C. Town, D. Grant, and R. Nelson, SSWAP: A Simple Semantic Web Architecture and Protocol for Semantic Web Services, *BMC Bioinformatics*, Vol. 10:309, 2009.

[60] V. Terziyan, and O. Kaykova, From Linked Data and Business Intelligence to Executable Reality, In: *International Journal on Advances in Intelligent Systems*, Vol. 5, Ns. 1&2, 2012, pp. 194-208.

[61] A. Katasonov, and V. Terziyan, Semantic Agent Programming Language (S-APL): A Middleware Platform for the Semantic Web In: *Proceedings of the Second IEEE International Conference on Semantic Computing (ICSC-2008) / International Workshop on Middleware for the Semantic Web*, August 4-7, 2008, Santa Clara, CA, USA, IEEE CS Press, pp. 504-511.

[62] Nikolov, A., Uren, V., & Motta, E. (2010). Data linking: Capturing and utilising implicit schema-level relations.

[63] T. Tiihonen, *How to Create Trust*, In the Presentation during the Tempus Information Days Helsinki, November 22, 2011, Available in: http://www.cimo.fi/instancedata/prime_product_julkaisu/cimo/embeds/cimowwwstructure/22799_TRUST_Tempus_Timo_Tiihonen.pdf .

[64] V. Terziyan, *TRUST: Towards Trust in Quality Assurance Systems. Brief Introduction of the Project Idea*, In the Presentation during the: TRUST (516935-TEMPUS-1-2011) Project Coordination Meeting, Kiev, Ukraine, October 20, 2011, Available in: http://www.cs.jyu.fi/ai/Quality-2.ppt .

[65] M. Klymova, *Ontology-Based Portal for National Educational and Scientific Resources Management*, In the Presentation during the: Universities Nationwide IT Days (IT-2007), Jyvaskyla, Finland, November 1, 2007, Available in: http://www.cs.jyu.fi/ai/OntoPortal-2007.ppt.

[66] *Web Site of the Ukrainian National Ontology-Based Portal for Management of Educational and Scientific Resources*, 2009-2012, Available in: http://ailab.kture.kharkov.ua/site/index.html.

[67] H. Muhleisen, T. Walther, and R. Tolksdorf, A Survey on Self-Organized Semantic Storage, *International Journal of Web Information Systems*, Vol. 7, No. 3, 2011, Emerald Group Publishing, pp. 205 – 222.