THE ACQUISITION AND SHARING OF DOMAIN KNOWLEDGE CONTAINED IN SOFTWARE WITH A COMPLIANT SIK ARCHITECTURE

Vasile Avram
Academy of Economic Studies (University in Economics), Bucharest, Romania
vasileavram@ie.ase.ro
http://www.avrams.ro
Abstract: The benefits and needs for extensive knowledge management towards the organizations require, among other things, to change software architectures for a better adaptation to the knowledge era. The change argued in this paper refers to adding a component in the form of one or many companion knowledge repositories expressed as Resource Description Framework (RDF), to systematically acquire, structure, formalize, store and maintain domain knowledge described as business rules, for all business rules that are incorporated in the software product itself. For every software adopting this architecture, the companion repositories acts as vectors to manage, communicate, transfer and share what is called “software inherited knowledge” (SIK), like a genetic repository. SIK architectures refer mainly to software that incorporates internally, mixed with its logic, the served domain business rules. It refers also partly to software that uses externalized business rules but keeps also business rules internally incorporated and does not refer at all to business rules management systems that define and manipulate externally all business rules. As it is defined, SIK allows applications to inherit domains rules from other applications and direct (or little adapted) usage of those business rules, from the Business Rules Management System (BRMS) the company may have. The inheritance of business rules, by using SIK architecture, is different from the inheritance induced by the internal software classes and objects, in object-oriented applications. One aim of this paper is to define and explain the SIK concept, together with all related concepts, from both theoretical and practical perspectives. Another aim of this paper is to exemplify how software can be designed or adapted to be compliant with SIK architecture. The last aim is to outline and emphasize the major benefits, for all users and environments where the application software is used, when adopting SIK architecture for software applications or when buying applications complying with that architecture.

Keywords: knowledge management; business rules; software inherited knowledge (SIK); software architecture.
1. Introduction

The competition pressure in the global market raised the demands for better quality, less cost production, and more accountability. The practice proves that all these are possible by an extensive use of knowledge. Intensive and extensive usage of knowledge creates for companies premises to have the ‘right product, at the right time, in the right place’. The knowledge resource becomes more and more critical to the survival and success of an organization in the global market since knowledge is situated at the core of the nowadays revolution in developing a knowledge economy.

The World Wide Web technologies have changed both the way interacting parties communicate with each other and the way business is conducted (Antoniou and van Hamerlan, 2008:1). The World Wide Web lies at the heart of the nowadays revolution in developing a knowledge economy or, generally, knowledge society in which knowledge resource is critical for both individuals and organizations. Knowledge management, which emerged initially as a ‘key activity for large business’ (Antoniou and van Hamerlan, 2008:3), becomes today a key activity for every kind of business. It becomes the way for preserving and better using the intellectual asset represented by the internal knowledge that the organization possesses. Considering that knowledge management concerns with ‘identifying, creating, representing, maintaining, and distributing knowledge for reuse, awareness, and learning across organizations’, consequently every organization must identify and exploit at least all its internal sources for knowledge.

2. Knowledge Management and Software Categories

In order for the knowledge to allow consistent application throughout an organization, it must be ‘documented, organized, published, revised, shared, and managed’ (CORTICON, 2009:2). The documented knowledge, that is explicit knowledge, represents the source for training employees and brings them to uniform standards in order to increase their implicit or ill-defined knowledge. Since the human language can induce uncertainty and create ambiguity, by its nature itself, the documented knowledge is almost all of the cases ‘inadequate for preservation, automation, and reuse of the intellectual capital’ (CORTICON, 2009:2). Knowledge uncertainty arises from the contradictory nature of knowledge given by its basic forms: absolute and relative. The migration from one to another is the basis for changes. The ambiguity can make policy (or procedure) application dependent on the implicit knowledge of the individuals applying the policies.

The knowledge can be documented by business people that posses it as formalized rule sets, by using descriptive languages and a declarative style of programming, and can be used later on to automate critical, recurring business
decisions, for example. Generally, formalized rules or production rules do not allow ambiguity and uncertainty, so they are ideal means for keeping the meaning of knowledge intact. The usage of a ‘declarative style of programming for describing business rules reduces considerably the maintenance problems but makes the algorithms extremely difficult to represent and flow control difficult to be followed and supervised by an application designer’ (Graham, 2007:75-76). Production rules are easy for humans to understand and, since each rule represents a small independent granule of knowledge, can be easily added or subtracted from a knowledge base. For this reason they have formed the basis for BRMS (Business Rules Management Systems) products that understand and act them as graphical representation, and become also machine understandable, represented as declarative descriptions, as used by the BRE (Business Rules Engine) and Semantic Web.

The increasing complexity of business relationships and global competition induces an increasingly complexity and dependence of organizations on Information Technology (IT). This complexity makes more and more important for organizations to use both business and IT best practices. Brand software, sold and delivered by leader IT companies, for business support such as Supply Chain Management (SCM), Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), Human Resources Information System (HRIS), Business Intelligence (BI) etc. incorporates good practices inspired from those of leader companies in various domains or, in other words, it incorporates the knowledge allowing the application and the operation of those practices. An organization adopting and using such software in its current operation, will start its activities and competitiveness at least from that point of good practices and associated knowledge.

In this context, regarding the used software, the knowledge management will have two aspects:

• Knowledge management, in the classic sense of knowledge about software product and its operation. This knowledge, acquired by training and practice, in time becomes also implicit for employees that uses the product;
• Knowledge management, in the sense of knowledge incorporated in the software products that is, generally, knowledge about the way software processes the business itself (as its business processes). This knowledge is generally documented externally in the product manuals and is a key factor in deciding a software adoption.

The software can be associated also with another knowledge category called ‘Architectural Knowledge (AK)’ (Aman and Babar, 2009:49) and represented by the knowledge surrounding software architecture design decisions, as described by Aman and Babar (2009). It includes the knowledge used in the knowledge intensive tasks executed during the system development steps to define, analyze,
model, develop, code etc the applications and is not the subject of this paper.

The knowledge incorporated in the software products and inherited by a company at software product adoption time is very difficult to be revealed to business people since, generally, this is included in the product documentation and revealed to the users when they are trained to use the software product. This knowledge becomes generally explicit / implicit for business people. More, almost all of the time, even if undesirable, there are a lot of differences between the documentation content, how this is understood by business people and how it is really implemented into the product.

Davenport and Prusak (1998:12-17) stipulates that “in a global economy, knowledge may be a company’s greatest competitive advantage” and consequently, the ability to survive and thrive comes only from the ability of the company to create, acquire, process, maintain, and retain old and new knowledge, facing complexity, uncertainty, and rapid changes. By integrating the knowledge inherited from software, the company can really prove its ability to exploit universally available knowledge. We can say about the knowledge embedded in software products that, when really used and exploited, it accelerates both the present and future work and has the same value as the past experience that the company used to accelerate its present and future work.

The knowledge embedded in software is mobile, similarly to the tacit knowledge of employees: the withdrawal of a software product or non-usage of all his incorporated knowledge in current activities (as unused branches or options) always leads to losing it, similarly to the way a company loses the tacit knowledge of a person who leaves it. The difference is that the unused options can be reconsidered and reintegrated in the company operation.

The incorporation in the company’s knowledge base of the knowledge contained by new adopted software products creates a real base for company’s model to migrate from a product-centered model to a knowledge-centered model.

Considering the way in which it incorporates and manipulates the modeled domain knowledge, the software will be classified in three broad categories of products:

- Business rules management systems (BRMS) that manipulate and execute business rules described, generally, as production rules. Production rules represented graphically and/or as declarative sentences are easy for humans to understand. Since each rule represents a small independent granule of knowledge it can be easily added or subtracted from a knowledge base. Because the rules are independent one from each other, they support the declarative style of programming which considerably reduces maintenance problems but makes algorithms extremely difficult to be represented and flow of control very hard to be supervised by a system designer. The rules formalism makes no allowance for uncertain knowledge. BRMS products can act any business rule described in terms of its rule description language and compliant with the constraints of the
domain deserved by that BRMS.

- “Externalized knowledge” products that externalize the business rules and knowledge and that allow changing some aspects of the behavior without requiring rewrite the application. Only externalized business rules and knowledge, that represents the source of problem, will be rewritten. This architecture gives a high flexibility and agility (but not the highest, as given by using BRMS) to the application, allowing it to adapt its behavior to the organization’s policies changes. This knowledge is described by a specific description language and understood and acted by an application that accepts a limited knowledge area.

- “Monolithic” products that incorporate business rules and knowledge together with the procedural part and the processing logic. They require rewriting the application if aspects of their behavior must change (triggered generally by business rules and knowledge changes). This architecture, even if it is very used in practice, is rigid to changes and requires rewriting the involved modules. More, they hide the knowledge in code and make it difficult to be used and even extracted by using different automated tools. The term monolithic is used here to outline that a software product (an entire application, a task or process, or even a service) incorporates all domain knowledge it models.

3. SIK Architecture

Since SIK (Software Inherited Knowledge) is proposed as an architectural component, we consider here the following two common definitions for software architecture:

- ‘Software architecture presents a view of a software system as components and connectors. Components encapsulate some coherent set of functionality. Connectors realize the runtime interaction between components’ (Albin, 2003: 3),

- ‘The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them’ (Bass, Clements, and Kazman, 2003:3).

Considering the previous definitions and the benefits and requirements for extensive knowledge management towards the organizations, the software architecture must change, doesn’t matter which software category it fits. The change refers to adding an architectural component, in the form of one or multiple knowledge repositories (Figure 1 a), in order to systematically acquire, structure, store and maintain knowledge, formalized as business rules for all domain business rules that are incorporated in the software product itself. The
formalization of the business rules is realized through RDF/XML serialized triples. The tool used to describe the RDF triples can be an RDF Parser that is able to convert the triples in the serialized format or an RDF Editor able to process directly the serialized format of the triples. This knowledge doesn’t refer to the knowledge used in the knowledge intensive tasks executed during the system development steps to analyze, model, develop, code etc the applications as defined in Babar and Gorton (2007:170). This knowledge refers to the domain knowledge of the business domain modeled by the application software and together with the knowledge about the operation of the current application, it represents the knowledge of interest to the end users of the application.

For every company adopting software having this architecture, the companion repository act as a vector for transport and communication of the knowledge inherited from software, for that reason the software companion knowledge repository was named ‘software inherited knowledge (SIK)’ (Avram, 2009:77). SIK must be accessible to all persons and/or applications having the rights to access and manipulate it. The knowledge involved here will be captured and maintained in the context of analysis, design, coding and testing phases (and of every reiteration of these required in implementation or current operation), and will be expressed in the form of business rules statements, as defined and understood by the software developer, since this incorporates the rules in the application body.

The documented knowledge, in the form of business statements or even sentences of a programming environment, will be transformed into ‘actionable, formalized, automatable software asset’ (CORTICON, 2009: 3). The formalized rules should be tested to identify ambiguity, incompleteness, redundancy and contradiction, to certify the validity and to ensure truly actionable business knowledge. The rules defined can be incorporated, eventual after an adaptation/translation to the language used by a BRMS for representing the business rules it can understand and act, into business applications and/or standard services enabling broad, reusable, scalable, automated distribution of these knowledge assets when and where they are needed.

Figure 1 b) illustrates an adaptation of the principle architecture of a Semantic Web Application as defined by Allemang and Hendler (2008:59-89) and the connection point with SIK repository. The RDF Parser of the Semantic Web Application will access the SIK Configuration file to be aware about the security policy applied. Later on, it will access the RDF documents and extract the business rules described here as RDF triples, will analyze and store them in the RDF Store from where, by using the RDF Query Engine to extract and combine them, the application will be able to analyze the stored knowledge and even to infer new knowledge. Other applications can access and exploit the repositories in a way similar to the one just described.

Since the sustainability of an organization and its success in front of the competition lie in its ability to manage and increase its stock of knowledge,
the SIK will act as a source of enriching with the contained knowledge every knowledge base deserving / assisting knowledge workers. The model chosen for SIK, in the form of a repository, will favor the ‘integrative architecture’ (Russ 2010:266) for KMS (Knowledge Management System).

Since one of the knowledge management cornerstones is to improve productivity by effective knowledge sharing and transferring the adoption of the SIK software architecture, at least for both “externalized knowledge” and “monolithic” software categories, which provide the knowledge companion containing the incorporated-knowledge, will be a source and solution for the knowledge extensive implementations and even for future automation.

The SIK can be visible to any authorized person, preventing the ‘black-box’ syndrome common to all monolithic applications. The user can learn and understand from the system behavior, making his interaction with the application easier.

The existing software can be adapted to the new architecture by adding its knowledge repository and by describing the business rules incorporated and/or by extracting them using the existing tools and methodologies, developed in the effort of converting existing systems into BRMS solutions.

The RDF repositories can be exploited and integrated by using the atomic functions and tools provided by open source libraries for Java, PHP, Ruby, Python etc., by using industry solutions such as Sesame, or can even indexed and searched by semantic search engines such as Yahoo! BOSS, Sindice, and Freebase. For knowledge representation, SIK uses the abstract model of RDF that allows ‘any application that understands the model to consume any data source using the model’ (Segaran, Evans, and Taylor, 2009:1).

4. Defining SIK

Since both the adoption of a standard language to define business rules and the adoption of a standard for Knowledge Interchange Format (KIF) appear to be too far in time, the proposal for description of the SIK is to use RDF (Resource Description Framework). The RDF will be used as shown in Berners-Lee et al (1999), Daconta, Obrst, and Smith (2003:85-117), Antoniou and Van Hamerlan (2008:25-112), and compliant with the specifications and recommendations of W3C (World Wide Web Consortium) for XML, RDF, and RDF/XML available at http://www.w3c.org. RDF has a simple data model, a formal semantic and a provable inference (Klyne and Carroll 2004). Because it uses an XML-based (eXtended Markup Language) syntax, it allows anyone to make statements about any resource. This format allows a better portability, being adapted to Web and used in intranet, Internet, and extranet for software applications integration and communications purposes. In all these environments, the software application can also be delivered as a service and SIK will allow for better communication of its capabilities.
XML plays the role of an interoperability mechanism and its use, primarily for data exchange between internal and external organizations, has achieved a greater maturity and adoption. By using an open standard syntax and verbose descriptions of meaning of data, XML is readable and understandable by everyone – not just by the application program and the person that produces it. XML Schema is a definition language that constrains XML documents to be compliant with a specific vocabulary and with a hierarchical specific structure. The XML Schema uses XML syntax to declare a set of simple or complex type declarations. For XML documents, the schema is just another document that resides physically somewhere, invoked as testimony and model for validity checks.

Daconta, Obrst, and Smith (2003:27-55) identifies four main characteristics of XML that make this successful, characteristics that are inherited by all other languages/products derived from XML. In the following paragraph three of these XML characteristics together with other characteristics of interest for the subject of this paper are described from an RDF (and sIK) perspective. RDF, being an XML-based language for describing resources, inherits the characteristics of XML, such as:

- The described resources are well-formed and valid. The well-formed is a mandatory requirement through which RDF complies with W3C syntax rules of XML. Validity has two components: the validity from the point of view of the used language (directly inherited from XML) and the validity from the point of view of knowledge management of business rules. The validation of RDF document is a critical component since the document is intended to be shared and processed by a large number and variety of applications. The SIK formalized rules should be tested to identify ambiguity, incompleteness, redundancy and contradiction, to ensure truly actionable business knowledge;

- It creates application-independent documents and data (Daconta, Obrst, and Smith 2003:28). A RDF document contains, in the same physical file, the document itself, both the document metadata (including document schema) and the data. An application that accesses the file is able to apply the structure described to corresponding data and to understand the meaning of those data and the data itself;

- It has a standard syntax and structure for metadata and data (Daconta, Obrst, and Smith 2003:28). They are described in the same physical document and use the same syntax rules as the ones in XML. The metadata, or data-about-data, allows associating meaning to those data, meaning which is understandable by both machines and humans;

- It allows both protection by encryption of the sensitive content of the document and mixing encrypted parts with unencrypted ones. Protection can be realized by using encryption mechanisms such as public keys or symmetrical keys, used in order to protect message exchange over the Internet. Through this protection SIK will be able to give proper access rights to processes and/or peoples;
• It allows describing multiple independent / dependent business rules in the same physical document. Having the possibility to describe both metadata and data in the same document, RDF can contain many namespaces and, in turn, each of these namespace can refer to a specific problem. The namespace, used as qualifier for the elements described, allows the sentences of a rule to spread across the document and, also, it allows those sentences to mix. Thus, if an application contains more branches, each one of them can have the rules described in the same RDF document even if they are variations of the same main rule;
• It is not a new technology (Daconta, Obrst, and Smith 2003:28). Being, in its linear representation, a language defined by using XML, it becomes an application of this one and inherits almost all of the rules of XML regarding the requirements for syntax, validity, usage, maintaining, and protection. Thus, it is an old technology used for a specific purpose.

RDF is not XML. It inherits all the XML characteristics but adds a very important characteristic: schema and data are both contained in the same document. A direct corollary of this characteristic is that we can mix together many business rules in the same RDF document, each one of them having possibly a different schema.

5. Benefits of SIK Adoption

By adopting the SIK proposed architecture for a software product, its consumers of this will have at least the following benefits:
• The members of the organization can have up-to-the-minute information about the business rules applied through the used software applications, in a formal and concise form that may help them improve business. This information can be extracted and even extended by using semantic search technologies and inference engines, respectively;
• Competitive analysis of products or requirements regarding the way to solve problems can have a real base: one can compare the business rules (and determine those that better fits his needs) and can require knowledgeably new approach;
• Automated reasoner can deduce (infer) conclusions from the knowledge given in SIK repositories, making the implicit like knowledge explicit like;
• Now there is a paradigm shift (or a strong migration) from proprietary stove-piped systems to open standards architectures. With the emergence of both, Cloud Computing and Semantic Web, the applications are materialized and delivered in the form of Web services and/or Software-as-a-Service (SaaS). More, since the next big trend in Web services is to be semantic-enabled, we can use information from Web services from different
organizations to perform correlations, aggregations, and orchestration. The service-driven technology is a response to the additional demand for agility and flexibility of company to respond in “market time”. The major industry trend is to realize an extensive knowledge management towards organizations as one of the responses to the effort to increase business agility and flexibility. This goal can be achieved by adopting, for Web service and SaaS, a SIK architecture that will offer the possibility to access directly the business rules described;

• With a growing knowledge base where ‘business rules, status reports, lesson learned, and competitive intelligence were all interconnected’, the competitiveness of the company can be really sustained by knowledge.

SIK can play a crucial role in describing business rules in a computer actionable format facilitating application software integration. Application can be sold as traditional or as knowledge repositories actionable via BRMS engines.

Corroborated with a Web service architecture for software, SIK can be one of the foundation for the next-generation information systems called ‘cooperative-information systems’ allowing an ‘intelligent’ mean to be dynamically integrated into customized and highly connected cooperative environments that exploit the Web services, and/or in fully automated environments that exploit the business rules repositories. From the point of view of the domain rules it models and acts, the application is transformed from an ‘isolated one’ into an ‘accessible one’.

SIK:

• Accelerates and automates the deployment and management of domain business rules as well-formed, well-designed, and valid domain knowledge. Since ‘knowledge management is particularly important for every organization having geographically dispersed departments’ (Antoniou and van Hamerlan, 2008:3), the knowledge repositories are the means for sharing and deploying that incorporated knowledge across these departments;

• Adds visibility to the way that current software defines and processes business rules and gives a real foundation for future software requirements;

• Creates the ability to integrate and unify different silos of knowledge. This ability is given by the fact that ‘RDF is based on domain-neutral model that allow one set of statements to be merged with another set of statements, even though the information contained in each set of statements may differ dramatically’ (Powers 2003:8). The repository model or integrative model is one of the models adopted by knowledge management systems (Russ 2010:266);

• Has the capability to provide knowledge management with the ability to respond in real time and to take action correspondingly. There are technologies and tools that allow analysis, transformation, and querying
of large amounts of RDF data;
• Uses the standards that ‘enable the Web to be a global infrastructure for sharing both documents and data, which make searching and reusing information easier and more reliable as well’ (Cardoso, 2007:3).

6. Conclusions

The main obstacle in providing better support to domain users is that, at present, the meaning of business rules content incorporated in software products is not human accessible. The business rules meaning is hidden to both users and machines. The documentation of the product itself is not enough for better understanding principles and acting ways. Of course, there are tools that can retrieve business rules, split them into parts and check for consistency but, when it comes to interpreting the rules and extracting information for users, the capabilities of current software are still very limited. These tools, that allows retrieving and extracting business rules from application software, do that in their own way and we can say that they act as a ‘tradutore-traditore’ (translator-traitor).

With a SIK adoption, the business rules become machine-accessible and human meaningful. This characteristic is given by the main goal of RDF to record knowledge in a machine-understandable format. The designer and modeler of business rules can describe them better and, later, we can use directly intelligent techniques to take advantage of these formalized representations.

SIK induces the benefits of organizing knowledge in conceptual spaces according to its meaning. The automated tools are able to extract new knowledge from repositories by exploiting the mechanisms provided by RDF that facilitates the combination of data. It also allows replacing the keyword-based search with query answering inside the company intranet and extranet and helps developing Semantic Web in a bottom-up approach (eRDF and RDF are supported formats for Yahoo! Slurp, the Yahoo! web robot that indexes and finds Web pages).

References

Aman-ul-haq and Babar, M. A. (2009) Tool support for automation architectural knowledge
extraction, University of Limerick Ireland, Shark’09, May 16, 2009 Vancouver, Canada, pp. 49-56.