Journal of Software Engineering and Applications, 2011, 4, 718-728
doi:10.4236/jsea.2011.412084 Published Online December 2011 (http://www.SciRP.org/journal/jsea)

A Knowledge Management Framework in Software Requirements Engineering Based on the SECI Model

Azeddine Chikh

Information Systems Department, College of Computer & Information Sciences, King Saud University, Riyadh, Saudi Arabia.
Email: az_chikh@ksu.edu.sa

Received October 7th, 2011; revised November 14th, 2011; accepted November 27th, 2011.

ABSTRACT

Software requirements engineering deals with: elicitation, specification, and validation of software requirements. Furthermore there is a need to facilitate collaboration amongst stakeholders and analysts. Fewer efforts were deployed to support them in performing their job on a day to day basis. To solve this problem we use knowledge management for software requirements engineering. This paper proposes a knowledge management framework, based on the SECI model of knowledge creation, aimed at exploiting tacit and explicit knowledge related to software requirements within a given software project. The core part of the proposed framework is a set of four sub systems "Socializer"; "Externalizer"; "Combiner"; and "Internalizer", attached to a couple of domain ontologies and a set of knowledge assets. Indeed we aim to facilitate a semantic based interpretation of knowledge assets related to software requirements by restricting their interpretation through the application domain and software requirements ontologies. We anticipate that this framework would be very helpful for stakeholders as well as analysts to exchange and manage their knowledge within a given software project. We show in the case study, through a virtual payroll project using the two-step approach: domain level requirements plus design level requirements, how the key elicitation SRE techniques are used during the first phase of domain requirements elicitation through the four subsystems of our framework.

Keywords: *Software Requirements Engineering, Knowledge Management, Domain Ontologies, SECI Model*

1. Introduction

The communities of software engineering and knowledge engineering share a number of common topics. Whereas software engineering research has been continuously struggling towards a higher abstract software modeling during the last decade, the knowledge engineering community has been enthusiastic to promote numerous modeling approaches to conceptualize a domain of knowledge. In the literature, several research works have been directed towards knowledge management in software requirements engineering (SRE) [1-4]. New era of SRE management starts focusing on knowledge management of software requirements within a given project. Such a knowledge must include not only the generic knowledge of individual specialty fields of the domain of SRE and the project application domain (payroll, finances, sales, etc.) but knowledge of the best practices captured from the previous and similar projects.

Activities in the domain of SRE typically involve people from at least two fields: 1) the business field (customers /users and other stakeholders) and 2) the IT field (analysts, requirements engineers and software project managers). This diversity of actors often produces important information flows and knowledge exchange that are difficult to manage. A broad variety of requirements engineering models, techniques, and technological environments such as Computer-Aided Analysis/Engineering tools have been designed and implemented by researchers. They are used to help gathering, analyzing, and documenting software requirements. However, a lack of efforts was observed in the way requirements engineering practitioners are being supported in their daily activeties. There is a need to help those actors managing collaboratively and exchanging their knowledge building shared practices (best practices).

After this introduction, Section 2 recalls the SRE domain. Section 3 introduces the use of ontologies in this domain. Section 4 explores the SECI model of knowledge creation and shows how it is applied to this domain.

In Section 5, we present a knowledge management framework in SRE based on the SECI model of knowledge creation. Then we integrate two domain ontologies to annotate the knowledge assets related to software requirements. Section 6 illustrates the use of the proposed framework through a case study related to a virtual software project in the domain of payroll. Finally the last section concludes the paper and points out directions for future work.

2. Software Requirements Engineering

The requirements for a system are the descriptions of what the system should do, the services that it provides and the constraints on its operation. The process of finding out, analyzing, documenting and checking these services and constraints is called requirements engineering (RE) [5]. SRE is a sub-category of (RE) that deals with the elicitation, specification, and validation of requirements for software [6] and it is critical for successful software development.

Domain analysis is the process through which an analyst learns background information. Some domains might be very broad, such as human resources. Others are narrower such as "Payroll". People who have a deep knowledge of a domain are called domain experts. Since the involved analysts are often not domain experts, they must learn about the problem domain from whatever sources of information. These include domain experts, books about the domain and existing software. The interviewing, observation, brainstorming and prototyping techniques can help with domain analysis [7].

The software requirements specification (SRS) is an official statement of what the system developers should implement. It should include a detailed specification of the system requirements [5]. It helps to understand what the system is supposed to do and how its use can support users and satisfy stakeholders. It is critical to the success of a development project. Accordingly it should be based on a model in which the result is an unambiguous and complete specification document. The generic IEEE standard 830-1998 [8] describes recommended approaches for the SRS. It describes the content and qualities of a good SRS. It can be adapted to the considered project.

Joint Application Design (JAD) and Participatory Design (PD) methodologies emphasize and promote cooperative work, among the various SRE' actors, in developing the SRS [9].

3. Using Ontologies in Software Requirements Engineering

The requirements on particular software are typically a complex combination of requirements from different stakeholders at different levels of an organization and from the environment in which the software will operate. Accordingly, the resulting SRS document should be clear, unambiguous, easy to understand, complete and consistent. In practice, this is difficult to achieve as stakeholders interpret the requirements in different ways and there are often inherent conflicts and inconsistencies in the requirements [5].

A different understanding of the concepts involved may lead to an ambiguous, incomplete specification and major rework after system implementation [10]. In addition, it is important to assure that all analysts and stakeholders in the analysis phase have a shared understanding of the concepts related to the application domain. Furthermore, even when users can express their needs, analysts find it difficult to write them accurately. The result is that the real needs, such as expressed by users, and the requirements, such as written in SRS, don't match.

SRE is concerned with interpreting and understanding stakeholders' terminology, concepts, viewpoints and goals. It aims to integrate these diverse views into a shared, correct and complete SRS. Therefore, it must concern itself with an understanding of beliefs of stakeholders (epistemology), the question of what is observable in the world (phenomenology), and the question of what can be agreed on as objectively true (ontology). Such issues become important whenever one wishes to talk about validating requirements, especially where stakeholders may have divergent goals and incompatible belief systems [11].

Gruber [12] defines ontology as a formal specification of a conceptualization. A conceptualization is an abstract simplified view of a domain that describes the objects, concepts and relationships between them that hold in that domain. Ontology can be used for both, to describe requirements specification [5,13] and formally to represent requirements content [1]. Further, the "domain model" can be described using an ontology language, with varying degrees of formalization and expressiveness. Ontologies seem to be well suited for an evolutionary approach to the specification of requirements and domain knowledge [1]. In addition, ontologies can be used to support requirements management and traceability [2].

Moreover semantic wikis are used in SRE process, as semantic and social-web based collaboration platforms by the diverse stakeholders participating in this process [14].

4. The SECI Model of Knowledge Creation

In the past, there have been a number of tools facilitating knowledge management (KM) activities, but they were not intended to explicitly integrate knowledge sharing and exchange. Most of the past KM research works used a typical top-down approach considering knowledge as a separate entity and focusing on the creation of central knowledge repositories that foster knowledge reuse and collaboration. Organizations have recognized that knowledge constitutes a valuable intangible asset for creating and sustaining competitive advantages. Knowledge sharing is an activity through which knowledge is exchanged among members of a community or an organization. Recent research on KM clearly recognizes the importance of sharing knowledge within organizations [15-17] to name but a few.

Researchers divide knowledge into two distinct forms: tacit and explicit. Polanyi [18] considers knowledge as either tacit or rooted in tacit knowledge. The type of knowledge that we have learned so well, and embedded in the unconscious mind, that we use them without thinking. Nonaka [19] argues that we can know more than we can tell. Indeed not all our knowledge can be expressed into objects, documents or artifacts. In 1991, Nonaka [16] adapted the definition of tacit knowledge of Polanyi and defined the explicit and tacit forms of knowledge.

- **Explicit knowledge (EK)** is easily expressed, captured, stored and reused. It can be conveyed as data and is accessible from databases, books, manuals and mail.
- **Tacit knowledge (TK)** is difficult to express and transfer to another person by means of writing it down or verbalizing it. It is deeply rooted in action and consists partly of technical skills and partly of mental models that we do not recognize in ourselves and cannot easily specify them.

Nonaka & Takeuchi [16] propose the SECI knowledge creation model as a key for KM in organizations. They view knowledge as an activity rather than an object and they focus on knowledge creation, collaboration and practices rather than knowledge transmission as stated in [20, 21].

The SECI model consists of three main elements: 1) four modes of knowledge conversion between the EK and TK; 2) a shared context called "Ba"; and 3) knowledge assets.

The creation of knowledge is a continuous process of dynamic interactions between EK and TK. The four modes of knowledge conversion interact in the spiral of knowledge, Socialization; Externalization; Combination; and Internalization as following:

- **Socialization:** sharing experience to create new TK;
- **Externalization:** articulating and converting TK to EK. TK becomes EK through metaphors, analogies, concepts, hypothesis, and models;
- **Combination:** restructuring and aggregating EK into new EK;
- **Internalization:** reflecting on EK and internalizing it into TK.

The "Ba" can be defined as the shared context where the four modes of knowledge conversion happen [22]. Naeve *et al.* [23] consider the Ba as "a place for interactive knowledge creation" (this space can be physical, virtual or mental). This Ba is divided into four contexts corresponding respectively to each knowledge mode: 1) **Originating Ba**: a space for social interaction; 2) **Dialoguing Ba**: a place where participants share and articulate their mental models and skills; 3) **Systemizing Ba**: a context where EK can be combined such as a collaborative environment; and 4) **Exercising Ba**: an internalization context where knowledge is exercised through action and practice.

The third element in the SECI model is the knowledge assets. They are defined by Nonaka [22] as a set of enterprise-specific resources, that are considered as inputs, outputs as well as moderating factors of the knowledge creation process and that are necessary to create values for the company. Four groups of knowledge assets are identified in [16] by Nonaka *et al.*:

- **Experiential knowledge assets** can be seen as hands-on experiences; skills acquired through dialogue, discussion and shared practice.
- **Conceptual knowledge assets** consist of EK articulated through images, symbols and language. These assets are based on the concepts held by members and stakeholders of the community.
- **Systemic knowledge assets** consist of systematized and packaged EK, such as explicitly stated technologies, product specifications, manuals, and documents.
- **Routine knowledge assets** consist of the TK that is customized and embedded in the actions and practices of the organization.

Based on SECI knowledge spiral model, Wan *et al.* [24] have put forward a knowledge creation model for software requirement development. Authors consider that knowledge dissymmetry is one of the forces that drive knowledge conversion. Therefore they argue that depending on experts on requirement, this model can reduce knowledge dissymmetry, and realize knowledge conversion and share more effectively and efficiently. Moreover Kamata [25] proposes the combination approach of KM and chance discovery in order to do better requirements definition. KM might be effective for shallow level. The author shows how the knowledge may flow through SECI process more sufficiently between the customer and the supplier, being respectively considered by the author as an application domain expert and an IT expert.

5. A Knowledge Management Framework in SRE Based on the SECI Model

The domain of SRE deals with a set of processes: elicitation; specification; and validation of software requirements. In software projects with high uncertainty, a better actors' participation increases the quality of SRS [26]. Therefore, SRE actors should collaboratively work to improve the quality of the SRS. We assume that this collaboration can be improved by using a knowledge management framework dedicated to SRE and based on SECI model of knowledge creation.

The remainder of this section describes the structure of the framework and is organized into three sub sections. The first two subsections describe how the domain ontologies and the SECI model are applied to SRE. Then the framework's architecture is proposed in the third subsection.

5.1. Applying Domain Ontologies to SRE

Our proposed framework aims to facilitate a semantic-based interpretation of software requirements and the related knowledge assets, within each SRE process, by restricting their interpretation through domain ontologies. Indeed, we advocate the idea that these domain ontologies can be used to describe software requirements related knowledge assets that flow through the SECI cycle of knowledge creation occurring during the SRE processes, thus providing them with a new dimension of content reusability.

Happel and Seedorf [10] differentiate between the use of ontologies in software engineering: 1) at development, such as using conceptual models of the problem domain and 2) at run-time, such as adaptations. According to this classification, our approach belongs to the second category.

Two domain ontologies are used by our framework to represent the domain knowledge: Application Domain Ontology (ADO) and Software Requirements Ontology (SRO).

- **ADO** involves understanding the application domain (e.g. payroll; finance; sales; library; etc.). In order to enable effective knowledge assets understanding, we have to further enhance semantics of their content. Therefore, we recommend that they should be further enhanced by providing application domain ontology based annotations of their content. Many specific application domain ontologies exist on the Web that can be found using Swoogle[1]—a semantic web search engine.
- **SRO** encompasses many SRE concepts. It covers many possibilities: requirements on various levels from goal-level to design-level, different requirements styles from plain text to diagramming, and from data requirements to quality requirements, many techniques and methods from elicitation to validation [27]. For SRO, we have selected SWORE[2] ontology (SoftWiki Ontology for Requirements Engineering) proposed by [28]. Despite the SWORE ontology describes only a small subset of the domain of SRE, our choice is motivated by its modular structure as well as its clear conceptual separation. However we have proposed an extension of SWORE by integrating further SRE concepts such as requirements styles and levels.

[1]http://swoogle.umbc.edu/.
[2]SWORE is available for download at: http://softwiki.de/SWORE.

Figure 3 visualizes an extract from the core of the SWORE ontology, which was developed in accordance with standards of the requirements engineering community [29]. Central to this ontology are the classes—Stakeholder and Abstract Requirement along with property details. Abstract requirements have the subclasses—Goal, Scenario, and Requirement each of which are defined by stakeholders and can be detailed by other abstract requirements. This enables the specification of abstract requirements at different levels of granularity. The collaborative aspects of requirements engineering are emphasized by integrating discussions amongst the stakeholders and voting in the model. In order to interlink requirements with existing documents or resources, SWORE contains the classes Raw Information and Reference Point together with suitable properties [28].

Style and Level, which are drawn using shading styles, are the main classes of the extended part. A software requirement might be specified in many styles (in plain text, as a diagram, as a table, screen, etc.). The goal design scale is composed of 4 software requirements levels: goal level; domain level; product level; and design level [27]. The goal-level requirement aims to justify why the customer wants to spend money on the product. It can be verified, although only after some period of operation. The domain-level requirement outlines the user tasks involved and requires support for these tasks. The product-level requirement specifies what comes in and goes out of the product (software). It identifies only the functions and features. The design-level requirement specifies one of the product interfaces in detail.

5.2. Applying the SECI Model to SRE

In our approach we consider the four SECI model modes as they occur in each SRE process, where the main actors are the analysts, users, managers, domain experts and other stakeholders. In **Figure 1** we adapt the SECI framework described in [23] to the context of SRE. Actors' individual knowledge will flow through the SECI cycle of knowledge creation. The knowledge tacit or explicit that is "input" to a given mode gets through a transformation process of dialogue, negotiation, conceptualization and agreement. Each SECI mode, as shown in **Figure 1**, occurs in a corresponding Ba.

Domain-related concepts and relationships of the domain ontologies, located at the center of the **Figure 1**, are used to annotate knowledge assets.

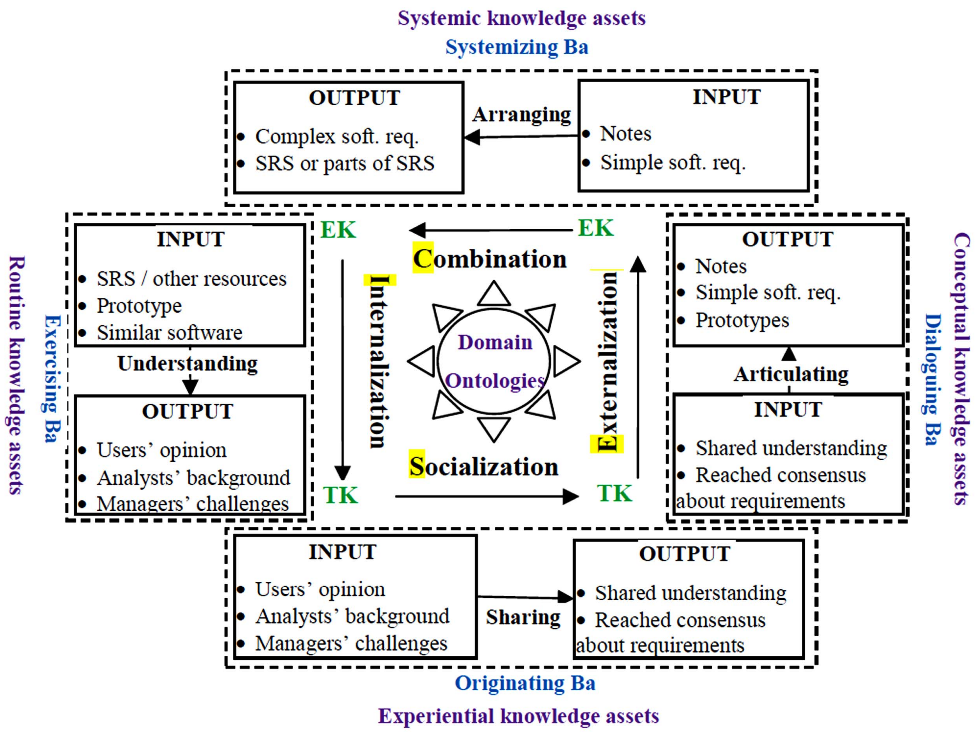

Figure 1. SECI Model adapted from Naeve *et al.* [23].

- **Socialization** mode occurs in the originating Ba. It is based on exciting experiences to learn from each other. TK is shared between SRE actors to have a better individual understanding of the software requirements, for instance: 1) Users might share their domain expertise, activities, preferences, problems and opinions; 2) Managers might share their challenges, strategies, constraints and objectives; 3) Analysts might share their background, technical points of view and solutions. One typical technique of socialization is JAD (Joint Application Design), a structured process in which users, managers and analysts work together for several days in a series of intensive meetings to specify or review system requirements. As an added plus, group members are more likely to develop a shared understanding of what the software is supposed to do, reaching consensus. The socialization process is controlled, among others, by the objectives, the moral values, the common background, and the readiness for collaboration of SRE actors; the balance of forces between these actors; and the means and rules of sociallization. At this point, we are in the stage of negotiation of meaning.
- **Externalization** mode happens in the dialoguing Ba. The preferences and wishes of users (TK) are articulated and conceptualized by analysts to produce for instance new (EK) such as notes or some specification using different styles (data dictionaries, entity-relationship diagrams, virtual windows for data requirements) or a prototype. Possible techniques of externalization are taking notes (during interviews or observation), brainstorming (during a close meeting) and prototyping. A software requirement specification such as a data requirement represented using a data dictionary style is one example of output of this mode. The externalization process is controlled, among others, by the objectives of analysts; the expression power of users in expressing their needs; and the means and rules of externalization. At this point we are in the stage of representation (reification) of knowledge.
- **Combination** mode occurs in the systemizing Ba. Separate and simple software requirements specification (EK) is further arranged into systematic and complex specification (EK). For example, user requirements and system requirements could be assembled to build an SRS. Possible techniques of combination are based on document reuse, building complex document from smaller ones by using transformation of structure. The combination process is controlled, among others, by the objectives of analysts and the means and rules of combination.
- **Internalization** mode occurs in the exercising Ba. Software requirements specification and other software requirements related knowledge assets (EK) are transformed into increased individual users and analysts understanding (TK). Moreover users can deduce new requirements (TK) through running a prototype (EK). Possible techniques of internalization are observation, active

reading (books about the domain, user documents, SRS, etc.) and using prototype or existing similar software. The internalization process is controlled, among others, by the objectives and the previous knowledge (background) of users and analysts and the means and rules of internalization.

The overall outcome from applying SECI model of knowledge creation to SRE is an increased and better EK and/or TK on software requirements among all the project actors: (analysts, users, managers, domain experts and other stakeholders). **Table 1** gives some examples of EK (pieces of knowledge that can be written down, transmitted, and understood by a beneficiary actor) and TK (chunks of knowledge that the users are not aware of or cannot express) in the domain of SRE. In practice most requirements are tacit. Analysts should specify only requirements which are not obvious. To do so, they must know what designers are likely to use these requirements for [27]. **Table 2** gives some examples of knowledge assets according to the four categories of SECI model of knowledge creation.

5.3. Framework's Architecture

The proposed framework is based on SECI model of knowledge creation. Moreover it uses ontologies to make semantic the different processes of this model. The system's architecture in **Figure 2** is drawn around four subsystems matching the four modes of SECI model and considered as their corresponding virtual "Ba". These subsystems are: "Socializer", "Externalizer", "Combiner", and "Internalizer". Furthermore, knowledge assets, contained within a knowledge repository, are annotated using concepts and relationships in domain ontologies. The functional details of the major components of the framework are explained as following:

1) The domain ontologies contain domain-related concepts and relationships in a structured and machine-interpretable format, and are used to annotate knowledge assets related to software requirements. We have considered two different ontologies (Section 5.1)—Application Domain Ontology (ADO) and Software Requirements Ontology (SRO). Semantic relations can model conflicts or dependencies between the requirements.

2) The knowledge repository is composed of knowledge assets either as instantiations (Is-a relationship) of ADO and SRO concepts or as related knowledge resources (Refers-to relationship). Indeed a knowledge asset might be a theoretical knowledge in the domain of SRE as well as a practical knowledge such as a best practice in the domain. Examples of knowledge assets are given in **Table 2**. Using a semantic representation, opposing to the traditional approach, aims to foster the understanding between SRE actors helping them to manage their knowledge assets related to software requirements. The framework allows indexing, using and reusing of these knowledge assets, based on concepts and relationships from ADO and SRO. Otherwise said, the knowledge assets related to software requirements become more definite, complete, consistent and suitable to share and reuse.

3) The four virtual "Ba"s, being virtual places for interactive knowledge creation, offers a set of complementtary tools and has as input and output either TK or EK

Table 1. Examples of EK and TK in SRE.

Category	Examples
Explicit knowledge	• A list or a textual description of functions to be provided by the software, a data dictionary, a prototype of the software screens
Tacit knowledge	• The ability to conduct a risk analysis is to identify risky areas of the software project and to decrease the risk. The hard part is to imagine the future work situations. This technique requires all sorts of knowledge and best practices that are not always known explicitly, even by expert analysts, and which are difficult to explicitly transfer to other analysts. As novice analysts (apprentices) learn how to conduct a risk analysis through simulation, observation, imitation and practice, they will learn new skills through on-the-job training. After a period of imitation and practice, a new analyst will not only deduce risks from procedures but also working with stakeholders and asking them which potential conflicts do they expect with other stakeholders? This turned out to be the secret for successful risk analysis.

Table 2. Examples of knowledge assets in SRE.

SECI knowledge assets categories	Examples of SRE knowledge assets
Experiential knowledge assets	Shared minutes during a JAD session
Conceptual knowledge assets	Separate requirements represented with different styles such as data dictionaries, entity relationship or data flow diagrams
Systemic knowledge assets	SRS or parts of it; Prototypes
Routine knowledge assets	A best practice to conduct an interview learnt from more experienced analysts

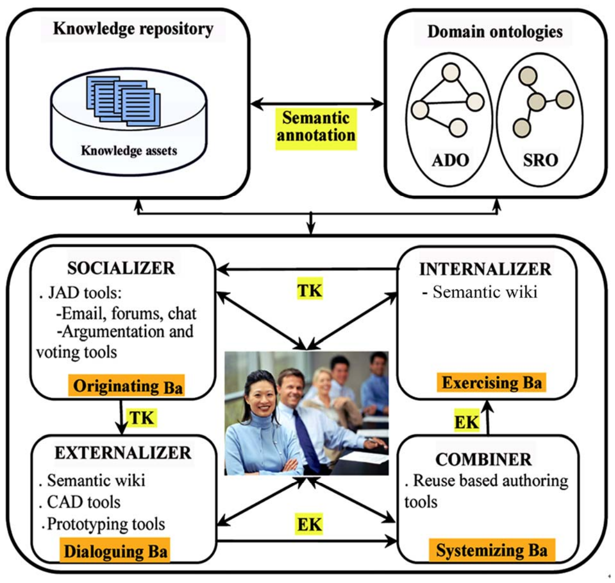

Figure 2. A knowledge management framework in SRE based on SECI model.

related to software requirements. Furthermore, a virtual "Ba" can be coupled to an Integrated Development Engineering (IDE) framework. Knowledge assets can be exported from a virtual "Ba" to the IDE framework. Accordingly, the rendering of business processes and system artifacts from the requirements description, can be partially automated [30].

a) "Socializer" offers a set of communication, argumentation and voting tools such as those used during JAD sessions. Having such a subsystem can help SRE actors to be more effective in guiding the socialization toward their goals. The exchanged TK, embedded in different discussions, might be linked to concepts and relationships from ADO and SRO helping to its understanding.

b) "Externalizer" offers a set of semantic tools such as semantic wikis, CAD tools, and prototyping tools, to quickly build a prototype. The TK as well as the resulted EK might be linked to concepts and relationships from ADO and SRO improving their semantic.

c) "Combiner" offers a set of reuse based authoring tools including fusion, merging or synthesis services. The primary EK as well as the composite EK might be linked to concepts and relationships from ADO and SRO improving their semantic.

d) "Internalizer" offers a set of active reading tools such as semantic wikis. Semantic annotation enriches the unstructured or semi-structured data with a context that is further linked to the structured knowledge of a domain. For this purpose, the semantic wiki exploits the concepts and the relationship, stored in the two domain ontologies ADO and SRO, and annotates the knowledge assets related to software requirements.

A semantic wiki is already proven to be a useful tool for the elicitation of requirements [31]. Among the existing semantic wikis, SoftWiki focuses on semantic collaboration with respect to software requirements activeties [32]. Its aim is to let large and distributed stakeholders be able to collect, semantically enrich, classify and aggregate software requirements. It is based on the Ontology (SWORE) adopted as SRO in our framework. Some features provided by Softwiki are listed in **Table 3**.

The approach used in this paper is more innovative than others used in similar works such as [24] and [25]. Indeed, it combines the SECI model and a couple of domain ontologies to make knowledge assets more accessible.

The knowledge repository contains a variety of assets such as software requirements and shared minutes during a JAD session (**Table 2**), offering a powerful source of reuse for analysts to build a new SRS or to complete an existing one. And finally the four subsystems include recent collaborative tools such as semantic wiki.

6. Case Study

We suppose a virtual project aiming to determine software requirements for a new payroll system. Examples of these requirements may include: the payroll processing should be completed within 24 hours; extra hours must be recorded separately from ordinary hours; management reports should contain detailed information about normal payment and extra-load payments for each department; management reports should allow managers to automatically analyze the data.

In order to illustrate the use of our framework in this project, we need first of all an application domain ontology (ADO) related to payroll domain, which will be used in concert with (SRO) in order to annotate the knowledge assets related to payroll software requirements. **Figure 3** shows a model of such an ontology [33]. The main concepts described are: Wage, Taxes and Employment contract. The employment contract can be terminable, not terminable or other. Terminable contracts can be seasonal or temporary.

In this project, we propose the use of the two-step approach proposed in [27]: domain level requirements plus design level requirements.

1) The first step is the domain level approach. In the basic version of this approach one primarily focuses on describing the user tasks collaboratively with users and collecting information on the data to be stored in the machine. The specification will contain the following elements: introductory parts including business goals; the limits of the system; data description; user tasks description saying that they have to be supported; a trace analysis;

Table 3. Softwiki's features.

Feature	Description
Adaptive content presentation	The content presentation of pages can change based on semantic annotations. Pages can be enriched with the display of semantically related pages in a separate box or information can be displayed, which is derived from the underlying knowledge base.
Enhanced navigation	Offering extra information about the relation a link describes. Furthermore, it can provide context-aware navigation.
Typing of annotation	Annotating links and text (requirements artifact) by giving them certain types.

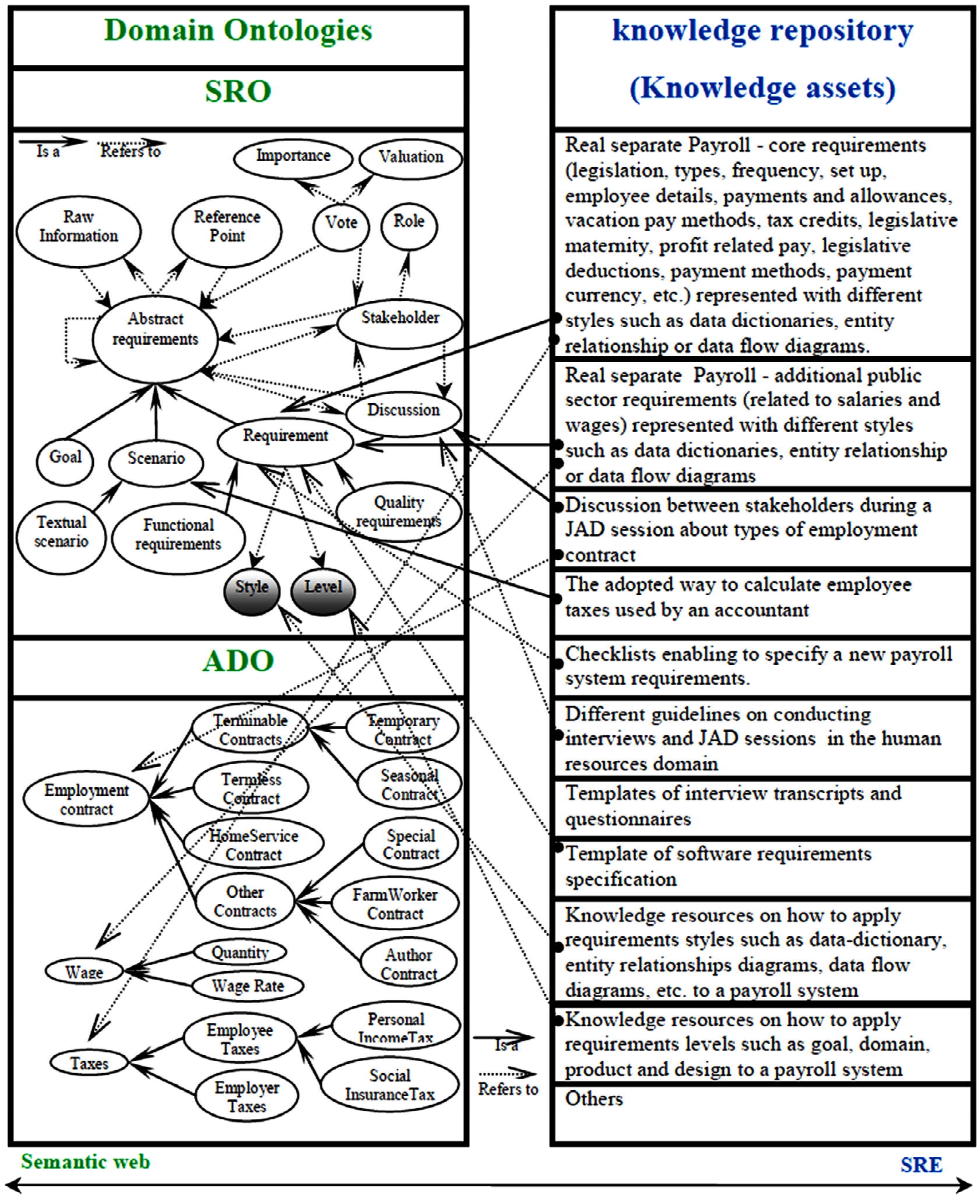
Domain Ontologies
SRO
Is a
Refers to
Importance
Valuation
Raw Information
Reference Point
Vote
Role
Abstract requirements
Stakeholder
Discussion
Requirement
Goal
Scenario
Quality requirements
Textual scenario
Functional requirements
Style
Level
ADO
Terminable Contracts
Temporary Contract
Employment contract
Termless Contract
Seasonal Contract
HomeService Contract
Special Contract
Other Contracts
FarmWorker Contract
Wage
Quantity
Wage Rate
Author Contract
Personal IncomeTax
Employee Taxes
Taxes
Employer Taxes
Social InsuranceTax
Is a
Refers to
knowledge repository
(Knowledge assets)
Real separate Payroll - core requirements (legislation, types, frequency, set up, employee details, payments and allowances, vacation pay methods, tax credits, legislative maternity, profit related pay, legislative deductions, payment methods, payment currency, etc.) represented with different styles such as data dictionaries, entity relationship or data flow diagrams.
Real separate Payroll - additional public sector requirements (related to salaries and wages) represented with different styles such as data dictionaries, entity relationship or data flow diagrams
Discussion between stakeholders during a JAD session about types of employment contract
The adopted way to calculate employee taxes used by an accountant
Checklists enabling to specify a new payroll system requirements.
Different guidelines on conducting interviews and JAD sessions in the human resources domain
Templates of interview transcripts and questionnaires
Template of software requirements specification
Knowledge resources on how to apply requirements styles such as data-dictionary, entity relationships diagrams, data flow diagrams, etc. to a payroll system
Knowledge resources on how to apply requirements levels such as goal, domain, product and design to a payroll system
Others
Semantic web
SRE

Figure 3. Use of domain ontologies in SRE within a virtual Payroll project.

and quality requirements for critical quality factors. A CRUD check (Create, Read, Update, and Delete) is necessary to see that all required data are handled in some task. It should be followed by a review. Stakeholders are primarily invited to correct the task descriptions.

2) The second step is the design level approach. A design task that creates design level requirements for complex interfaces. A prototype of screens is developed on the basis of task description and data description. Then the prototype is reviewed and validated through a usability test to see its effective support of the user tasks. Finally, verification consists to see that design-level requirements are satisfied.

Figure 3 illustrates the use of the domain ontologies (ADO) and (SRO) and the knowledge repository, as two components of the framework, within the payroll system project. Payroll system related knowledge assets from the knowledge repository are connected to concepts from both ADO and SRO domain ontologies according either "refers to" or "is a" links.

Furthermore, we assume that the above two-step approach follows an iterative process using a spiral model whose main output is the SRS. The combination of the two levels (domain and design) with the three SRE processes (elicitation; specification; and validation) results in six phases as shown in **Figure 4**: (1) domain requirements elicitation; (2) domain requirements specification; (3) domain requirements validation; (4) design requirements elicitation; (5) design requirements specification; and (6) design requirements validation. The number of iterations will change according the importance and the context of each project.

Table 4 shows how the key elicitation SRE techniques interviewing; observation; and questionnaires are used during the first phase of domain requirements elicitation, through the four subsystems of our framework: "Socializer", "Externalizer", "Combiner", and "Internalizer".

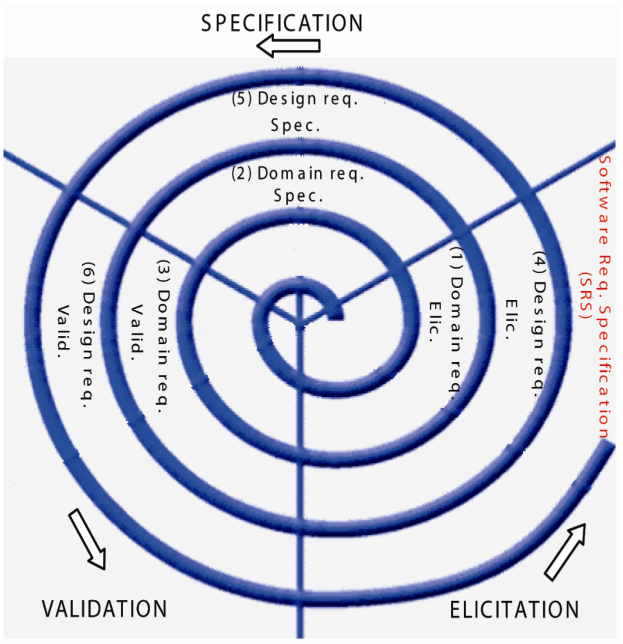

Figure 4. A spiral view of the iterative two-step approach process.

Table 4. Using the framework in the two-step approach during the domain requirements elicitation phase.

	Interviewing	Observation	Questionnaires
Socializer (S)	(S) will help to foster discussion between payroll stakeholders offering communication tools such as emailing, forums, chatting, etc.		
Externalizer (E)	(E) should offer to • analysts a prototyping tool to help them to translate in real time the users' needs into prototype screens • interviewees a semantic wiki helping the interviewee to express their demands	(E) should offer to the analysts a video recording tool	(E) should provide to stakeholders web based questionnaires to gather their opinions and suggestions about some specific aspects of the current payroll system
Combiner (C)			(C) should provide to analysts statistical tools that transform elementary data into more elaborated data
Internalizer (I)	(I) should offer to analysts a semantic wiki to annotate the interviews transcripts	(E) should offer to analysts a semantic wiki to annotate the observation notes	(I) should offer to analysts a semantic wiki to annotate the questionnaires results

7. Conclusions

In this paper, we aim to enable advanced services for all key actors of the software requirements processes (elicittation; specification; and validation). We came up with the idea of applying SECI as a knowledge creation model to these processes. We have developed a knowledge management framework in SRE based on SECI as a formalization of this idea. This framework integrates four subsystems ("Socializer", "Externalizer", "Combiner", and "Internalizer") matching the four modes of SECI model and their corresponding "Ba". These subsystems are attached to the repository either for creation or exploitation of knowledge assets or both. Furthermore they are connected to a couple of domain ontologies ADO and SRO to foster semantic research.

The main novelty of our proposed framework is to reconsider the SRE activities according a knowledge based approach combining SECI model of knowledge creation and domain ontologies.

The proposed framework is still at a conceptual level. Currently, we are working on developing a prototype to analyze the framework's effectiveness for real-life applications. As a future plan, we intend to connect this framework to a reuse based software requirements authoring framework. The idea is to support the knowledge management process according to SECI model by reusing similar knowledge assets from past software projects.

REFERENCES

[1] B. Wouters, D. Deridder and E. Van Paesschen, "The Use of Ontologies as a Backbone for Use Case Management," *Proceedings of the European Conference on Object-Oriented Programming ECOOP, Workshop: Objects and Classifications, a Natural Convergence*, Sophia Antipolis and Cannes, 2000.

[2] V. Mayank, N. Kositsyna and M. Austin, "Requirements Engineering and the Semantic Web, Part II. Representation, Management, and Validation of Requirements and System-Level Architectures," *ISR Technical Report*, University of Maryland, Baltimore, 2004

[3] J. Lasheras, R. Valencia-García, J. T. Fernández-Breis and A. Toval, "Modeling Reusable Security Requirements Based on an Ontology Framework," *The Journal of Research and Practice in Information Technology*, Vol. 41, No. 2, 2009, pp. 119-133.

[4] H. Kaiya and M. Saeki, "Using Domain Ontology as Domain Knowledge for Requirements Elicitation," *Proceedings of the* 14*th IEEE International Requirements Engineering Conference*, Minneapolis/St. Paul, 11-15 September 2006, pp. 189-198.

[5] I. Sommerville, "Software Engineering," Pearson Education, Upper Saddle River, 2011.

[6] P. Bourque and R. Dupuis, "Guide to the Software Engineering Body of Knowledge," IEEE Computer Society, Washington DC, 2004.

[7] D. D. Champeaux, D. Lea and P. Faure, "Object-Oriented System Development," Addison-Wesley Longman Publishing Co., Inc. Boston, 1993.

[8] IEEE Std. 830-1998, "IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society," *Software Engineering Standards Committee*, 20 October 1998, No. SH94654.

[9] D. Herlea, "Users Involvement in Requirements Engineering Process," *Proceedings of the* 10*th Annual Knowledge Acquisition Workshop KAW*96, Banff, 6-14 November 1996.
http://ksi.cpsc.ucalgary.ca/KAW/KAW96/herlea/FINAL.html

[10] H. J. Happel and S. Seedorf, "Applications of Ontologies in Software Engineering," *Proceedings of the international Workshop on Semantic Web Enabled Software Engineering SWESE*, Athens, 5-9 November 2006.
http://km.aifb.kit.edu/ws/swese2006/final/happel_full.pdf

[11] B. Nuseibeh and S. Easterbrook, "Requirements Engineering: A Roadmap," *Proceedings of the International Conference on Software Engineering ICSE*, Limerick, 4-11 June 2000.
http://mcs.open.ac.uk/ban25/papers/sotar.re.pdf

[12] T. R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *International Journal Human-Computer Studies*, Vol. 43, No. 5-6, 1995, pp. 907-928. doi:10.1006/ijhc.1995.1081

[13] B. H. C. Cheng and J. M. Atlee, "Research Directions in Requirements Engineering, Future of Software Engineering FOSE," *Proceedings of ICSE, IEEE Computer Society*, Minneapolis, 2007, pp. 285-303.

[14] B. Hoenderboom and P. Liang, "A Survey of Semantic Wikis for Requirements Engineering," Technical Report RUG-SEARCH-09-L03, SEARCH, University of Groningen, Groningen, 2009.

[15] P. Hildreth and C. Kimble, "The Duality of Knowledge," *Information Research*, Vol. 8, No. 1, 2002.

[16] I. Nonaka and H. Takeuchi, "The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation," Oxford University Press, New York, 1995.

[17] A. Cabrera and E. F. Cabrera, "Knowledge-Sharing Dilemmas," *Organization Studies*, Vol. 23, No. 5, 2002, pp. 687-710. doi:10.1177/0170840602235001

[18] M. K. Smith, "Michael Polanyi and Tacit Knowledge, the Encyclopedia of Informal Education," 2003.
http://www.infed.org/thinkers/polanyi.htm

[19] E. Wenger, R. McDermott and W. M. Snyder, "Cultivating Communities of Practice," Harvard Business School Press, Boston, 2002.

[20] M. A. Chatti, R. Klamma, M. Jarke and A. Naeve, "The Web 2.0 Driven SECI Model Based Learning Process," *Proceedings of the* 7*th International Conference on Advanced Learning Technologies ICALT*-2007, Niigata, 18-20 July 2007, pp. 780-782.

[21] C. Hoadley and P. G. Kilner, "Using Technology to Transform Communities of Practice into Knowledge-Building Communities," *SIGGROUP Bulletin*, Vol. 25, No. 1, 2005, pp. 31-40.

[22] I. Nonaka, R. Toyama and N. Konno, "SECI, Ba and Leadership: A Unified Model Knowledge Creation," *Long Range Planning*, Vol. 33, No 1, 2000, pp. 5-34. doi:10.1016/S0024-6301(99)00115-6

[23] A. Naeve, P. Yli-Luoma, M. Kravcik and M. D. Lytras, "A Modeling Approach to Study Learning Processes with a Focus on Knowledge Creation," *International Journal Technology Enhanced Learning*, Vol. 1, No. 1-2, 2008, pp. 1-34. doi:10.1504/IJTEL.2008.020228

[24] J. Wan, H. Zhang, D. Wan and D. Y. Huang, "Research on Knowledge Creation in Software Requirement Development," *Journal of Software Engineering & Applications*, Vol. 3, 2010, pp. 487-494.

[25] M. I. Kamata, "Software Requirements Elicited through Human-Centric Chance Discovery," *AAAI Fall Symposium Technical Report FS*-02-01, 2002, pp. 99-105.

[26] K. E. Emam, S. Quintin and N. H. Madhavji, "User Participation in the Requirements Engineering Process: An empirical Study," *Requirements Engineering Journal*, Vol. 1, 1996, pp. 4-26. doi:10.1007/BF01235763

[27] S. Lauesen, "Software Requirements: Styles and Techniques," Addison-Wesley, Boston, 2002.

[28] T. Riechert, K. Lauenroth, J. Lehmann and S. Auer, "Towards Semantic based Requirements Engineering," *Proceedings of* 7*th International Conference on Knowl- edge Management I-KNOW*'07, Graz, 5-7 September 2007, pp. 144-151.

[29] K. Pohl, "Requirements Engineering," Dpunkt Verlag, Grundlagen, Prinzipien, Techniken, 2007.

[30] T. Riechert and T. Berger, "Leveraging Semantic Data Wikis for Distributed Requirements Elicitation," *Proceedings of the* 4*th Workshop on Wikis for Software Engineering Wikis*4*SE at* 31*st International Conference on Software Engineering ICSE*'09, *IEEE Computer Society*, Vancouver, 19 May 2009, pp. 12-19.

[31] B. Decker, E. Ras, J. Rech, P. Jaubert and M. Rieth, "Wiki-Based Stakeholder Participation in Requirements Engineering," *IEEE Software*, Vol. 24, No. 2, 2007, pp. 28-35. doi:10.1109/MS.2007.60

[32] S. Lohmann, P. Heim, S. Auer, S. Dietzold and T. Riechert, "Semantifying Requirements Engineering—The Softwiki Approach," *Proceedings of the* 4*th International Conference on Semantic Technologies I-SEMANTICS*, ACM, Graz, 3-5 September 2008, pp. 182-185.

[33] J. Trinkunas and O. Vasilecas, "A Graph Oriented Model for Ontology Transformation into Conceptual Data Model," *Information Technology and Control*, Vol. 36, No. 1A, 2007, pp. 126-132.