A Reference Model for Knowledge Management in Software Engineering

J. Andrade, J. Ares, R. García, S. Rodríguez, and S. Suárez

***Abstract*— Software engineering is a discipline in which knowledge and experience, acquired in the course of many years, play a fundamental role. In this discipline changes are particularly fast, and new methodologies, techniques and technology constantly appear and modify or refine the existing knowledge.**

In this context, this work proposes the use of a knowledge management strategy that will allow each person involved in software development to access the best possible available knowledge at the right time. Specifically as a first step towards introducing a knowledge management programme that supports the software process, we present a formalization scheme that is able to represent and, therefore, capture and transmit the relevant organizational knowledge and metaknowledge that exists in software development. In this way, we facilitate the transmission of experiences and knowledge inside the organization and make institutional learning possible. Finally, our proposal will be illustrated by means of its application to software requirements engineering.

***Index Terms*— Corporate memory, knowledge descriptors, knowledge management, software engineering, viewpoint-based requirements engineering.**

I. Introduction

Since software engineering (SE) is a typical knowledge-intensive discipline that evolves very fast and involves a large number of people, different phases and different activities [1], it is one of the disciplines that can benefit most from knowledge management (KM) [2]. In this context, a KM programme could allow us to capture, explicit and institutionalize the knowledge and experience related to each software development project, and thus provide the organization with considerable benefits [3]. More precisely, the adequate use of KM to SE would allow:

1) To share experiences of individual workers (or groups) and institutionalize these experiences through the years, making each worker act almost like an expert.
2) To establish a common modus operandi at organizational level, helping the organization to efficiently use the same methodologies, techniques and technologies.
3) To facilitate the adoption and transfer of new methodologies, techniques and technologies between different departments or even partners.

A successful KM project, be it in the field of software development or any other field, must necessarily observe three key aspects. The first aspect is the implantation of the necessary mechanisms that guarantee a total involvement of the employees in the KM programme [4]; i.e., establish an organizational culture based on sharing and collaborating. The second aspect is to facilitate the exchange of the tacit or implicit knowledge among the employees, regardless of their physical and temporal location [5]. The third and last aspect, which supports the two previous ones, is the use of a corporate memory, as a means for description and interchange of relevant knowledge [6].

The corporate memories combine two repositories that, depending on the type of knowledge they contain, could be called corporate knowledge base or yellow pages:

1) The first repository compiles the relevant organizational knowledge with the aim of providing it to the employees whenever, wherever and in the format required. This repository includes two clearly differentiated elements: the knowledge itself (for instance, best practices) and metaknowledge (i.e. lessons learned, from the terminology used by Weber et al [7]). The first element is already institutionalized knowledge. For an organization that develops software, for instance, this element would describe the development process that is implemented at the organizational level. The second element is knowledge aimed to refine the first. An example of a knowledge chunk that would fit in this category is the adaptation that each expert has performed on the organizational process, on the basis of his/her knowledge and experience. This second element receives the generic name of metaknowledge and can be classified into positive experiences, negative experiences and false maneuvers [8].
2) The second repository corresponds to the identification

Manuscript received March 10, 2006. This work was supported in part by the Dirección Xeral de Investigación e Desenvolvemento da Xunta de Galicia (Autonomous Government of Galicia - project PGIDIT03PXIA10501PR), and the University of A Coruña.

J. Andrade is with University of A Coruña, Facultad de Informática. Campus de Elviña, s/n. 15071. A Coruña. Spain (e-mail: jag@udc.es).

J. Ares is with University of A Coruña, Facultad de Informática. Campus de Elviña, s/n. 15071. A Coruña. Spain (e-mail: juanar@udc.es).

R. García is with University of A Coruña, Facultad de Informática. Campus de Elviña, s/n. 15071. A Coruña. Spain (e-mail: rafael@udc.es).

S. Rodríguez is with University of A Coruña, Facultad de Informática. Campus de Elviña, s/n. 15071. A Coruña. Spain (Corresponding author: phone: +34 981 16 70 00; fax: +34 981 16 71 60; e-mail: santi@udc.es).

S. Suárez is with Technical University of Madrid, Facultad de Informática, Campus de Montegancedo, s/n. 28660. Boadilla del Monte, Madrid. Spain (e-mail: ssuarez@infomed.fi.upm.es).

and notification of the human and nonhuman elements that are sources of additional knowledge, i.e., that possess knowledge not included in the previous repository. Its importance lies in the fact that a KM programme should not try to embrace all knowledge and all lessons learned that exist in the organization. Since such a pretension is not realistic, this repository will at least point to knowledge that cannot be included but, nevertheless, may be useful.

Many approaches have recently arisen to define and introduce KM programmes in an organization [9]. None of the current proposals, however, responded to the expectations of the developers of this type of programmes. This is mainly due to the fact that they approach the above aspects in a prescriptive manner: they indicate the steps that must be taken without describing how to proceed [9]. For instance, those proposals do not clearly advise on how to define an organizational knowledge base, or which elements to consider in it; aspects that are crucial in this type of programme.

This situation is caused by the fact that the research in the KM field focused on the study of the management process (the M) rather than on the object that is going to be managed, i.e. the knowledge itself (the K). Only the proposal of Wiig et al. [10] identifies a small set of descriptors that support the explicitation—formalization—of the knowledge, but (i) this identification is not the result of an exhaustive study of the knowledge, and (ii) it does not allow a complete formalization, because it is limited to certain descriptive characteristics.

In order to contribute to ameliorate this situation, and as a step previous to, and independent of, the application of KM to software development, we intend to refine those current generic proposals by making them more descriptive. To achieve this descriptive orientation, it is essential that we study the object that is going to be managed, and thus base the definition of the management phases on the basis of knowledge classification and characteristics. Therefore, this paper has its basis on the study of knowledge as a means to define the structure of the organizational knowledge base and to articulate the construction of a KM programme from a knowledge descriptive approach. Our definition considers the knowledge of the organizational environment as well as the associate metaknowledge. Both elements will follow the formalization scheme that we propose here.

In order to demonstrate the above, this paper is organized as follows: Section II presents the proposed formalization scheme for organizational knowledge base; Section III shows the evaluation of the proposed scheme through its application to software requirements engineering; and, finally, Section IV presents the main conclusions of this study.

II. SCHEME OF KNOWLEDGE FORMALISATION

As it has been mentioned at the previous section, the study of knowledge should be specially borne in mind in order to achieve a descriptive proposal for KM. Considering this, the authors have performed a strict study with the aim of determining those types of knowledge that should be formalized and conceptualized within any KM system. This study that, has been detailed in [11], is based on:

1) The five conceptualization hypotheses, which allow us to identify and manage the concepts that exist in any domain [12]: abstraction, contraposition of a system of concepts with the real world, connection between a system of concepts and a system of language, expression of concepts by nonsyncategorematic terms, and need for set theory.
2) The formal conceptualization definition, which establishes the generic abstraction for any conceptualization of any domain. In this sense, conceptualization is formally defined as a (C, R, F) triplet [13]. This triplet includes, respectively, the concepts (C) presumed or hypothesized to exist in the world—universe of discourse—; the relationships (R), in the formal sense, between concepts—relational basis set—and the functions (F), also in the formal sense, defined on the concepts—functional basis set.
3) Natural language, which is considered the means par excellence used by human beings to transmit knowledge in any domain.

As a result of this study, we reached the conclusion that every piece of knowledge can be classified into one of those levels [11] [14]:

– Static: Constituted by the structural or declarative knowledge in a particular domain. In other words, true facts about the domain that can be used in operations: concepts, properties, relationships and constraints.
– Dynamic: Constituted by the behavior that takes place in the domain, that is, functionality, action, process or control, like inferences, calculations and step sequences. This level can be further subdivided into two sublevels:
  - Strategic. Includes what to do, when to do it and in which sequence.
  - Tactical. Specifies how to obtain new declarative knowledge.

Fig. 1 shows these levels and their interrelationships. As the figure shows, the strategic level manages the tactical level—arrow (1) in Fig. 1— by specifying which inferences and calculations are needed at each nonatomic step. Moreover, the other two levels manage the declarative level, as it contains knowledge that is used for decision-making on the face of different alternatives or bifurcation points that affect a step sequence—arrow (2) in Fig. 1—and under which basis the inferences and calculations are made—(3) in Fig. 1. The pyramidal diagram shows that the functional levels of knowledge are related by a support structure. The strategic knowledge is located at the top, as it controls how the problem is dealt with. On the other hand, the tactical knowledge needs the declarative knowledge, which, furthermore, is the most abundant. Therefore, the declarative level has been placed at the base of the pyramid with the tactical level above it.

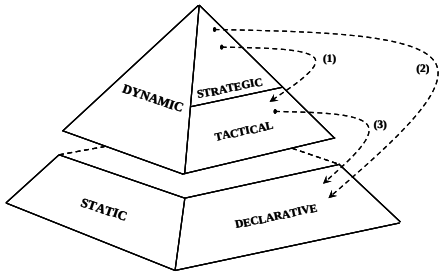


Fig. 1. Functional levels of knowledge and their interrelationships.

This taxonomy allows us to conceptualize and formalize the knowledge (K) of any domain in general [14], and therefore of the SE in particular, in order to proceed to its subsequent management (M). Henceforth, to formalize the functional knowledge levels that were identified, we segmented the formalization scheme in as many subschemes as levels. Thus, there is a static subscheme—which consists of the declarative level—and a dynamic subscheme—which consists of the strategic and the tactical levels.

This partition initially seems particularly useful in SE, since this field has not only process-oriented knowledge and domain-oriented knowledge but also technical knowledge.

Each element, or knowledge chunk, in each of these sublevels can be described by means of a set of descriptors that are shown in the next subsection. Although we identified three different subschemes, we also establish a subset of common descriptors that characterize any knowledge asset and do not depend on each knowledge level.

Last, we would like to point out that the above-mentioned metaknowledge is knowledge about knowledge and, therefore, knowledge. By this reason, our scheme is also applicable to metaknowledge. In other words, a lesson learned can be conceptualized and formalized on the basis of the three knowledge levels presented here.

A. Common descriptors

The descriptors that are common to every knowledge asset, at any knowledge level, are divided into terminological, qualificatory and relational descriptors. Terminological descriptors allow us to clarify the nomenclature used to make reference to any knowledge asset in the organization. Qualificatory descriptors characterize each knowledge asset from the viewpoint of its utility. Finally, relational descriptors allows us to find more details of the knowledge asset, if needed, either pointing to an additional knowledge source (yellow pages) or following already existing links between knowledge and its associated metaknowledge.

The terminological descriptors are the following:

– [Name]: The main term by which the asset is known in the organization.
– [Synonyms]: Other terms by which the asset may be known.
– [Abbreviations]: Abbreviations that are used for the asset.
– [Observations]: Any type of clarification concerning the used terminology.

The qualificatory descriptors are:

– [Topic]: Theme(s) to which the asset deals with.
– [Reliability]: Level of trust that should be put on the asset.
– [Impact]: Relevance of the asset in the domain of interest (business area, organizational process, etc.).
– [Validity]: Time window in which the asset is considered valid.
– [Availability]: Time frame in which the asset is available.
– [Security]: Organizational roles that have access to the asset.
– [Language]: Language of the asset.

Finally, relational descriptors are:

– [Support sources]: Sources—human and, or, nonhuman—that contain additional knowledge for better understanding the asset. This descriptor allows us to integrate a yellow pages system in a simple manner. The authors propose a detailed description of the formalization of a yellow pages system in [15].
– [Associate knowledge/Lessons learned]: If we are formalizing a lesson learned, this descriptor indicates the knowledge that is being refined. If the asset is a "basic" knowledge asset—strategic, tactical or declarative—, this descriptor indicates the related lessons learned that exist in the KM programme.
– [Producing agent]: Agent that provides the knowledge asset. This descriptor allows us to put into operation the mechanisms that recognize and guarantee each knowledge supplier and thus cover the first key aspect that was mentioned in the introduction: a culture of exchange and collaboration.

B. Strategic subscheme

The strategic subscheme formalizes the step sequence needed for a certain organizational function; that is to say, it defines what to do, when, and in which order.

Since a KM programme must provide knowledge to various access profiles (coordinators and executors), we formalized the complete decomposition of a function into its constitutive steps, considering both the nonterminal (decomposed) steps and low-level atomic steps (nondecomposed).

Descriptors for decomposed steps are:

– [Description]: Description of the strategic step in terms of the coordinator.
– [Decomposition structure]: Description of the decomposition associated to this step, considering the substeps into which it is decomposed.
– [Execution structure]: Description of the execution order of the substeps into which the step is decomposed.
– [Preconditions]: Previous requisites that are necessary to carry out the step. They agree with the preconditions of the constitutive substeps.
– [Postconditions]: Output requisites that are necessary to finalize the execution of the step. They are coherent with the postconditions of the constitutive substeps.
– [Inputs]: Elements—concepts, relationships and properties—that are necessary for the execution of the step. The list of inputs depends on the inputs of the constitutive substeps.

– [Outputs]: Elements—concepts, relationships and properties—that are generated as a consequence of the step's execution. The list of outputs depends on the outputs of the constitutive substeps.
– [Control elements]: Elements—concepts, relationships and properties—that are implied at each bifurcation point of the execution order. This descriptor allows us to reflect relationship (2) in Fig. 1.
– [Coordinator]: The organizational role that is responsible for the coordination of the step.
– [Observations]: Any related observation, e.g. the justification of the need to execute the step, or its purpose.

The nondecomposed steps are formalized by the following descriptors:
– [Description]: Description of the step in terms of the executor.
– [Preconditions]: Necessary previous requisites.
– [Postconditions]: Requisites to finalize the step.
– [Inputs]: Elements—concepts, relationships and properties—that are necessary to execute the step. These elements are determined by the Operational mode, which is the descriptor of the associated tactical knowledge.
– [Outputs]: Elements—concepts, relationships and properties—that are generated by the execution of the step. These elements are determined by the Operational mode, which is the descriptor of the associated tactical knowledge.
– [Operational mode]: Reference to the tactical knowledge that obtains the outputs from the inputs. This descriptor allows us to reflect interrelationship (1) in Fig. 1.
– [Executor]: Organizational role charged with the execution of the step.
– [Observations]: Any related observation, e.g. the final purpose of the step.

C. Tactical subscheme

This subscheme formalizes knowledge that indicates the way in which a certain task should be carried out. It is connected to the nondecomposed steps—the task it is associated with—defined in the strategic subscheme.
– [Description]: Description of the considered tactical knowledge.
– [Basic elements]: Elements—concepts, relationships and properties—that are required to proceed with the definition. This descriptor allows us to reflect part of interrelationship (3) of Fig. 1.
– [Conclusion elements]: Elements—concepts, relationships and properties—that are obtained by the definition. This descriptor allows us to complete the formalization of interrelationship (3) of Fig. 1, partially formalized by the previous descriptor.
– [Definition]: Algorithm, mathematical expression, inference or procedure that describes the considered tactical knowledge.
– [Observations]: Any related observation, e.g. limitations in the application of the definition.

D. Declarative subscheme

This section presents the descriptors that formalize the conceptual elements at the declarative level: concepts, relationships and properties. Constraints identified in the previous section are incorporated as part of the element they affect, i.e. in the properties and the relationships.

The descriptors that are used for the formalization of the concepts are the following:
– [Description]: Description of the concept.
– [Properties]: Characteristics that describe the concept.
– [Relationships]: Associations maintained with other concepts.
– [Observations]: Any related observation, e.g. the justification of the need of the concept, or its purpose in an entrepreneurial function.

The descriptors used for formalizing the relationships are the following:
– [Description]: Description of the identified relationship.
– [Elements]: Concepts that take part in this relationship.
– [Type]: The type of the relationship: generalization/specialization, aggregation, instantiation or domain defined (see [11]).
– [Properties]: The characteristics that describe the relationship.
– [Constraints]: Limitations to which the relationship is subject, i.e. restrictions that are applicable to the occurrences of the concepts that can participate in the relationship.
– [Observations]: Any related observation, such as the relevance of the relationship in the organizational operation considered.

For formalizing the properties, the following descriptors were established:
– [Description]: Description of the property considered, be it the property of a concept or of a relationship.
– [Element]: Concept or relationship to which the property is linked and which allows its description.
– [Data category]: Type of values of the property and, if necessary, the measure and precision units that are required.
– [Range]: Allowable values taken by the property.
– [Constraints]: Limitations of the property, both in terms of its own nature (e.g. compulsoriness or optionality) and its possible values (e.g. dependency on the value of other properties).
– [Source]: The human or nonhuman source that provides the values assigned to the property.
– [Observations]: Any related observation, e.g. a value by default or the justification of the need of the property or of its purpose.

E. Formalization strategy

The strategy we propose in order to obtain the relevant knowledge involves the elaboration of the knowledge pyramid (Fig. 1) from top to bottom. The first step goes from the strategic level to the tactical one (relationship labeled (1)). As strategic and tactical knowledge are being identified, the

relevant declarative one can be obtained (relationships (2) and (3)). Given that "relevant" means "declarative knowledge directly related with strategic and tactical knowledge", a more focused bottom-up search should be done at the pyramid. Of course, the pyramid shape has not been arbitrarily chosen, since it reflects that there are fewer elements at the top (strategic level) than at the basis (the declarative level).

This order is, of course, only an approximation, as each case will have its own particularities. Additionally, there was no clear difference; we only indicate what naturally emerged predominantly.

For each identified asset of knowledge, the proposed formalization scheme will help to find a generic and exhaustive set of descriptors. Nevertheless, there will always be a necessary adaptation of the scheme to each particular domain or situation (like, for instance, SE).

Once the knowledge base is complete, the lessons learned that appear in the course of time must be gathered, codified in the proposed scheme and incorporated into the repository.

In this way we can obtain a KM system where knowledge and metaknowledge are unified under the same framework.

III. EVALUATION OF THE FORMALISATION SCHEME

The practical evaluation of the defined scheme will be shown through its application to a particular domain within SE, more concretely the Silva's method for detection, classification and resolution of discrepancies among requirements [16] [17]. This method is currently being applied to an entrepreneurial environment [17]. By this reason, the knowledge acquisition process was greatly simplified, so the work was mainly focused on the proposed scheme for the formalization of the relevant knowledge. We consider, however, that the scheme will be also applicable to other areas of SE, and even with less difficulty than requirements engineering (RE), as RE deals with more socio-technical and "soft" difficulties that other more technical areas like design, testing, etc.

The method itself can be resumed in few words: Some proposals, named as Viewpoint-Based RE (VBRE), focused on the study and analysis of the multiplicity of stakeholder viewpoints, have emerged in the RE field (a survey has been published on the subject [18]). These proposals consider that there is a multiplicity of stakeholders that take part in a requirements process, inevitably leading to discrepancies (conflicts and inconsistencies) among them. The two main points of VBRE are that (i) discrepancies are not undesirable, as they can be used to improve the elicitation of requirements [19] [20] and (ii) discrepancies require (not necessarily immediate) actions to be handled. Systematic VBRE approaches need to properly diagnose each discrepancy found, before making any decision about what to do next. This diagnosis includes not only discrepancies location and identification, but also their classification. Once diagnosed, discrepancies should be handled accordingly [19].

This new approach to VBRE [16] started from the assumption that management of discrepancies could be enhanced when taking into account the different categories of statements managed during the requirements process, in order to properly compare one statement against another. Categorization of those statements into domain knowledge, requirements, interface descriptions, etc. can help the requirements engineer at the time of comparing them and classifying the discrepancies found, hence speeding up the VBRE process. For categorizing the statements we used the KSR model [21], widely known in the RE community. Our assumption was empirically tested in [17] and a software tool was built for helping process management, in order to facilitate its industrial uptake.

The following sections show the formalization that resulted from the application of the proposed KM scheme to the selected domain. We also present an example of the formalization of a lesson learned.

A. *Formalization of the knowledge*

Next, and to give an example, we present several fragments of the knowledge base that resulted from application of the process shown at section II.E. Some aspects have been simplified because they are too extensive and because the purpose of this section is not to describe discrepancy management, but to show the application of the proposed formalization scheme.

Table I presents the formalization of the decomposed strategic step "Discrepancy detection and resolution." We observe how knowledge contained in [16] that is associated to this step is represented according to the descriptors that were defined in section III.

TABLE I. DECOMPOSED STRATEGIC STEP "DISCREPANCY DETECTION AND RESOLUTION"

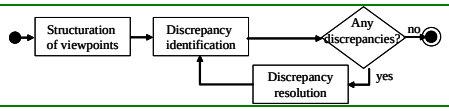
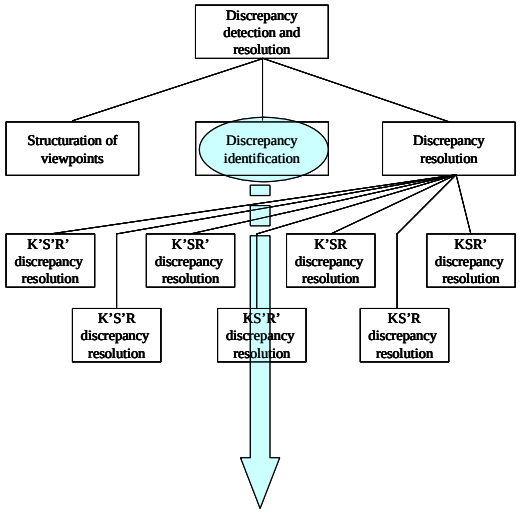
Description	Process by which the discrepancies between two given viewpoints are resolved.
Decomposition structure	Decomposition tree shown in Fig. 2.
Execution structure	
Preconditions	Having realized the process of requirements eduction on various viewpoints.
Postconditions	There is no discrepancy between the considered viewpoints.
Inputs	Statement, Equivalence class, Discrepancy criterion, Overlap criterion.
Outputs	Statement.
Control elements	Discrepancy.
Coordinator	Requirements engineer.
Observations	VBRE is based on the fact that several actors, that consider the problem from different perspectives or viewpoints, intervene in any requirements process. Since there are several viewpoints, they can be discrepant. The identification of these situations provides a way of anticipating errors from the early development phases, when the cost of resolving them is much smaller.

Table I shows that, in order to detect and resolve discrepancies, we must carry out three activities: structure the viewpoints, identify the existing discrepancies and, finally, resolve them. Fig. 2 shows how to synthesize (represent)—on the basis of the defined descriptors—the knowledge associated to the nondecomposed strategic substep "Discrepancy identification", and how to formalize the associated tactical knowledge.

Fig. 3 shows the formalization of the concept "Specification"—one of the key elements in the domain—and of its recursive relationship "Discrepancy." It also illustrates the representation of the property "Discrepancy Criterion", which is a property of the previous relationship.



Nondecomposed strategic step Discrepancy identification

Description	Step that identifies the existing discrepancies between two viewpoints in order to direct their subsequent resolution.
Preconditions	The viewpoints that must be analyzed were structured according to the KSR model
Postconditions	All the existing discrepancies were identified and classified.
Inputs	K, S, R, Equivalent class, Discrepancy criterion, Overlap criterion.
Outputs	Discrepancy.
Operational mode	Discrepancy detection and classification.
Executor	Requirements engineer.
Observations	VBRE proposes an early identification of the discrepancies, since they will allow us to direct the eduction process. However, the early identification of the discrepancies does not imply that they must be resolved immediately.
Support sources	KSR Model [21]

Tacit Knowledge Discrepancy detection and classification

Description	Procedure that identifies the discrepancy types and the discrepant statements that exist between two viewpoints.
Basic elements	K, S, R, Equivalence class, Discrepancy criterion, Overlap criterion.
Conclusion elements	Discrepancy.
Definition	1. Obtain K (K_i U K_j), S (S_i U S_j) and R (R_i U R_j) 2. For each equivalence class of S: 2.1. Obtain the kernel of the discrepancy (S_{ker}) 2.2. If it is discrepant in S, check if the discrepancy originates in K: 2.2.1. Obtain the overlap of S_{ker} and K 2.2.2. Obtain the kernel of the discrepancy (K_{ker}) 2.2.3. If it is discrepant in K, check if the discrepancy has repercussions on R: 2.2.3.1. Obtain the overlap of K_{ker} and R 2.2.3.2. Obtain the kernel of the discrepancy (R_{ker}) 2.2.3.3. If it is discrepant in R, the discrepancy type is K'S'R' The discrepant statements are: S_{ker}, K_{ker} and R_{ker} 2.2.3.4. If it is not discrepant in R, the discrepancy type is K'S'R The discrepant statements are: S_{ker} and K_{ker} 2.2.4. If it is not discrepant in K, check if the discrepancy in S has repercussions on R: 2.2.4.1. Obtain the overlap of S_{ker} and R 2.2.4.2. Obtain the kernel of the discrepancy (R_{ker}) 2.2.4.3. If it is discrepant in R, the discrepancy type is KS'R' The discrepant statements are: S_{ker} and R_{ker} 2.2.4.4. If it is not discrepant in R, the discrepancy type is KS'R The discrepant statements are: S_{ker} 2.3. If it is not discrepant in S, check if there are discrepancies in K: 2.3.1. Obtain the overlap of the partition of S and K 2.3.2. Obtain the kernel of the discrepancy (K_{ker}) 2.3.3. If it is discrepant in K, check if it has repercussions on R: 2.3.3.1. Obtain the overlap of K_{ker} and R 2.3.3.2. Obtain the kernel of the discrepancy (R_{ker}) 2.3.3.3. If it is discrepant in R, the discrepancy type is K'SR' The discrepant statements are: K_{ker} and R_{ker} 2.3.3.4. If it is not discrepant in R, the discrepancy type is K'SR The discrepant statements are: K_{ker} 2.3.4. If it is not discrepant in K, check if there are discrepancies in R: 2.3.4.1. Obtain the overlap between R and the elements of K overlapped with the partition of S 2.3.4.2. Obtain the kernel of the discrepancy (R_{ker}) 2.3.4.3. If it is discrepant in R, the discrepancy type is KSR' The discrepant statements are: R_{ker} 2.3.4.4. If it is not discrepant in R, there is no discrepancy
Observations	The order that is established for the detection of discrepancies (first S, then K and finally R) is based on the following heuristic considerations: 1. When starting with S, we first identify the discrepancies that affect the interface of the system, which assures us that we start with discrepancies that are relevant to the system. 2. When continuing with K, we obtain descriptive information, whose discrepancies may provoke discrepancies in R. Once the discrepancies in K are located, we can therefore trace their consequences into R.

Fig. 2. Navigation through the knowledge "Discrepancy identification."

Concept Specification

Description	Description of the interface between the future system and its environment.
Properties	NA
Relationships	It is a statement (generalization/specialization relationship), Discrepancy. K and S determine R.
Observations	The specification of the system can only take place when we know the phenomena of the domain that were used for the construction of the K and R sets. The requirements engineer, together with the user, should delimit the borders of the environment and the system, by identifying the phenomena that may be shared with the machine, and the way in which this partition will take place (of course, several possibilities may indeed exist).
Abbreviations	S.

Relationship Discrepancy

Description	Situation in which, due to the consideration of a set of statements, other statements cannot be considered. This situation is determined by a discrepancy criterion.
Elements	K, S, R.
Type	Defined by the domain.
Properties	Discrepancy criterion, Discrepancy Type, Discrepancy overlap.
Constraints	This relationship is only meaningful between statements of the same type (e.g., a statement that belongs to K can only be discrepant with another statement of K).
Observations	The term discrepancy that is used here includes the concepts of inconsistency and conflict in the terms used by Finkelstein et al [20] and Easterbrook et al [22]: – Inconsistency: Occurs when a rule that establishes a necessary relationship between two elements is broken. For instance, if all children below a given age are supposed to attend school and one child does not follow this rule. – Conflict: Occurs when the goals of one party interfere with the goals of the other, that is, what one party wants is not shared by the other. For instance, if the tutor of a child opposes to its schooling, there are conflicting interests between the State and the tutor, and a conflict is generated.

Property Discrepancy criterion

Description	This criterion determines whether two or more statements are discrepant.
Element	Discrepancy.
Data category	NA (depends on the type of notation used to express the viewpoints).
Range	NA.
Constraints	Compulsory.
Source	Requirements engineer.
Observations	This criterion depends on the type of notation used in the eduction process. We can also use different criteria as the development progresses, e.g., in early stages we can use a more flexible criterion, which becomes more restrictive in the course of the process.

Fig. 3. Navigation through the concept Specification.

The formalization scheme proposed, as it should, a generic and exhaustive set of descriptors. Nevertheless, there is always a necessary adaptation of the scheme to each particular situation. In this application, specifically, we adapted the terminological, qualifying and relational descriptors: given that we wanted to avoid unnecessary detail in the process and the result, we only used those descriptors that were required at any given moment. For example, the descriptor "Support sources", annexed to the nondecomposed strategic step "Discrepancy identification" in Fig. 2, was used to clarify the bibliographic reference that describes the KSR model (in other words, [21]).

B. *Formalization of the metaknowledge*

Once the knowledge base is complete, those lessons learned that appear in the course of time must be gathered and incorporated into this repository. As mentioned in section II, the proposed scheme is perfectly apt for this task, since metaknowledge is just knowledge that refines already existing knowledge.

Table II shows an extract from a lesson learned which is directly related with the previously mentioned tactical knowledge. In this Table, the effect of the lesson learned on the already formalized knowledge is highlighted. This lesson originated from the suggestion of a requirements engineer who, in the third cycle of the application of the discrepancy detection and classification process, realized that she was revising nondiscrepant knowledge for the second time. Past discrepancies were resolved and, after that, the engineer wondered why she should revise statements whose discrepancies had already been resolved in previous cycles. This lesson provoked the refinement of the previously presented tactical knowledge, and the appearance of a new property—"Marked"—associated to the concepts K, S and R.

TABLE II. EXTRACT OF THE LESSON LEARNED ABOUT DISCREPANCY DETECTION AND CLASSIFICATION

…	…
Definition	1. If we wish to analyze again all the affirmations, including those that were already considered, obtain: K (K_i U K_i), S (S_i U S_i) and R (R_i U R_i) 2. If we do not wish to analyze again the already considered affirmations, obtain: K as the set of unmarked statements of (K_i U K_i), S as the set of unmarked statements of (S_i U S_j), and R as the set of unmarked statements of (R_i U R_j) 3. For each equivalence class of S: **…**
Observations	1. The order that is established for the detection of discrepancies (first S, then K and finally R) is based on the following heuristic considerations: **…** 2. Since the detection and classification of discrepancies takes place in several cycles, it is adequate not to contemplate statements related to discrepancies that were resolved in previous cycles (marked).

The lesson learned is connected to the knowledge it refines by the descriptor "Associate knowledge/Lessons learned." We also incorporated the descriptor "Producing agent" pointing to the engineer that suggested the lesson. This feature carries out the reward policy for knowledge contribution [6]. Similarly, we associated the tactical knowledge "Discrepancy detection and classification" with this lesson through its descriptor "Associate knowledge/Lessons learned."

IV. CONCLUSIONS

This paper has presented a generic formalization scheme that responds to general knowledge needs that exist in any domain, and in software engineering in particular. This scheme particularly focuses on the definition of the structure of an organizational knowledge base; a repository that, together with the yellow pages, constitutes one of the most relevant elements of a KM programme: the corporate memory.

In order to obtain this scheme, we analyzed the object that has to be managed (the knowledge), its types, and its descriptors. Starting from (i) the formal definition of a conceptualization, (ii) the hypotheses of conceptualization and (iii) the natural language, we obtained a formalization scheme of the knowledge by considering its types (functional taxonomy) and its defining characteristics (descriptors).

The proposed scheme allows to:

1) Create a formal and systematic definition of the organizational knowledge base. This situation avoids the *ad hoc* definition of a corporate memory.
2) Establish the necessary basis to adequately acquire, formalize and transmit the knowledge and experiences that exist in the organization, with the aim of obtaining an institutional learning.

In the particular case of software development organizations that was exposed, we noted that the application of the scheme has provided the following improvements:

1) Facilitate the adaptation of the software process, in this case by considering innovative research results. This adaptation (i) was facilitated in time and shape (ii) was uniformly assumed and applied, and (iii) was globally transmitted to the involved software engineers.
2) Give support to the continuous improvement and adaptation of the defined software process (cf. Table II), considering the knowledge contributions of the software engineers, who really furnish the best knowledge assets thanks to their experience.

It also should be noted that the proposed scheme is not only useful for formalizing a corporate memory, as it also serves as a basis for defining KM support tools. In the present case, we defined a tool that currently support knowledge management related to the above viewpoint-oriented method and that is described in [23]. Moreover, this scheme allows describing and directing the activities of a KM initiative. In fact, the proposed scheme has allowed us to articulate the most relevant phases of a descriptive KM methodological framework, whose first version is described, applied and evaluated in [8]. Particularly,

the phases that most benefit from the scheme are knowledge acquisition and assimilation (conceptualization and representation), as can be concluded from the example provided here.

Finally, it should be highlighted that not all the aspects of this experience have been so positive, since not always is easy to involve the workers for providing their expert knowledge. This intrinsic situation to any KM system shows their need for being complemented with programmes that motivate knowledge sharing [24].

ACKNOWLEDGMENT

We would like to thank Valérie Bruynseraede (Research Transfer Office of the University of A Coruña) for her invaluable help in translating this paper.

REFERENCES

[1] I. Rus, and M. Lindvall, "Knowledge management in software engineering," *IEEE Software*, vol. 19(3), 2002, pp. 26-38.

[2] J. S. Edwards, "Managing software engineers and their knowledge," in *Managing Software Engineering Knowledge*, A. Aurum, R. Jeffery, C. Wohlin, and M. Handzic, Eds., Berlin: Springer-Verlag, 2003, pp. 5-27.

[3] A. Birk, T. Dingsoyr, and T. Stalhane, "Postmortem: never leave a project without it," *IEEE Software*, vol. 19(3), 2002, pp. 43-45.

[4] T. H. Davenport, and L. Prusak, *Working Knowledge: How Organizations Manage What They Know*. Boston: Harvard Business School Press, 2000.

[5] K. Wiig, *Knowledge Management Foundations: Thinking about Thinking. How People and Organizations Create, Represent and Use Knowledge*. Texas: Schema Press, 1993.

[6] G. Van Heijst, R. Van Der Spek, and E. Kruizinga, "Corporate memories as a tool for knowledge management," *Expert Systems with Applications*, vol. 13(1), 1997, pp. 41-54.

[7] R. Weber, D. W. Aha, and I. Becerra-Fernandez, "Intelligent lessons learned systems," *Expert Systems with Applications*, vol. 17, 2001, pp. 17-34.

[8] S. Rodríguez, *Un Marco Metodológico para la Gestión del Conocimiento y su Aplicación a la Ingeniería de Requisitos Orientada a Perspectivas*. PhD Thesis, Department of Information and Communications Technologies, University of A Coruña. 2002.

[9] B. Rubenstein-Montano, J. Liebowitz, J. Buchwalter, D. Mccaw, B. Newman, and K. Rebeck, "A systems thinking framework for knowledge management," *Decision Support Systems*, vol. 31, 2001, pp. 5-16.

[10] K. Wiig, R. de Hoog, and R. Van Der Spek, "Supporting knowledge management: a selection of methods and techniques," *Expert Systems with Applications*, vol. 13(1), 1997, pp. 15-27.

[11] J. Andrade, J. Ares, R. García, S. Rodríguez, and A. Silva, "Human-centered conceptualization and natural language," in *Encyclopedia of Human Computer Interaction*, C. Ghaoui, Ed., Hershey: Idea Group, Inc., 2006, pp. 280-286.

[12] J. A Díez, and C. U. Moulines, Fundamentos de Filosofía de la Ciencia. Barcelona: Ariel S. A. 1997.

[13] J. Ares, and J. Pazos, "Conceptual modelling: an essential pillar for quality software development," *Knowledge-Based Systems*, vol. 11, 1998, pp. 87-104.

[14] J. Andrade, J. Ares, R. García, J. Pazos, S. Rodríguez, and A. Silva, "A Methodological Framework for Generic Conceptualization: Problem-Sensitivity in Software Engineering," *Information and Software Technology*, vol. 46 (10), 2004, pp. 635-649.

[15] J. Andrade, J. Ares, R. García, S. Rodríguez, and S. Suárez, "Lessons learned for the knowledge management systems development," in *Proceedings of the 2003 IEEE International Conference on Information Reuse and Integration*, W. W. Smari, Ed., Las Vegas: IEEE Systems, Man and Cybernetics Society, 2003, pp. 471-477.

[16] A. Silva, "Requirements, domain and specifications: a viewpoint-based approach to requirements engineering," in *Proceedings of the Twenty-Fourth International Conference on Software Engineering*, W. Tracz, Ed., New York: ACM Press, 2002, pp. 94-104.

[17] J. Andrade, J. Ares, F. J. López, J. Pazos, S. Rodríguez, and A. Silva, "Computer-assisted Discrepancy Management. A Case Study in Research Transfer to Industry," *Journal of Research and Practice in Information Technology*, vol. 36(4), 2004, pp. 295-315.

[18] P. Darke, and G. Shanks, "Stakeholder viewpoints in requirements definition: A framework for understanding viewpoint development approaches," *Requirements Engineering*, vol. 1(2), 1996, pp. 88-105.

[19] B. Nuseibeh, S. Easterbrook, and A. Russo, "Leveraging inconsistency in software development," *IEEE Computer*, vol. 4, 2000, pp. 24-29.

[20] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh, "Inconsistency handling in multiperspective specifications," *IEEE Transactions on Software Engineering*, vol. 20(8), 1994, pp. 569-578.

[21] P. Zave, and M. Jackson, "Four dark corners of requirements engineering," *ACM Transactions on Software Engineering and Methodology*, vol. 6(1), 1997, pp. 1-30.

[22] S. Easterbrook, and B. Nuseibeh, "Using viewpoints for inconsistency management," *Software Engineering Journal*, vol. 11(1), 1996, pp. 31-43.

[23] M. Seoane, *Desarrollo de una Herramienta de Soporte a la Gestión Explícita del Conocimiento Organizativo*. BSc Project, Department of Information and Communications Technologies, University of A Coruña. 2003.

[24] G. P. Huber, "Transfer of knowledge in knowledge management systems: unexplored issues and suggested studies," *European Journal of Information Systems*, vol. 10(2), 2001, pp. 72-79.

Javier Andrade. Dr. Andrade is Associate Professor in the Information and Communications Technologies Department at the University of A Coruña, Spain. His research interests in computer science include conceptual modeling, knowledge management and natural language processing. He worked as software engineering and technological solutions consultant at IAL Software Engineering and Norcontrol Soluziona (Quality and Environment Department). He has a B.S. and Ph.D. in computer science. He is author of several book chapters and publications in software engineering.

Juan Ares. Dr. Ares is Director of the Information and Communications Technologies Department at the University of A Coruña, Spain. He is Associate Professor and co-Director of the Software Engineering Laboratory at this University. His research interests in computer science include conceptual modeling, knowledge management and software process assessment. He worked as director and consultant in several organizations, including Norcontrol Soluziona and Arthur Andersen. He has a B.S. and Ph.D. in computer science. He is editor of several books and author of numerous chapters and publications in software engineering.

Rafael García. Dr. García is Director of the Computer Training Unit at the University of A Coruña, Spain. He is Associate Professor and co-Director of the Software Engineering Laboratory at this University. His research interests in computer science include conceptual modeling, knowledge management and project management. He was project leader in several organizations, including Quibus Computers and Sistema Base. He has a B.S. and Ph.D. in computer science. He is editor of several books and author of numerous chapters and publications in software engineering.

Santiago Rodríguez. Dr. Rodríguez is Associate Professor in the Information and Communications Technologies Department at the University of A Coruña, Spain. His research interests in computer science include conceptual modeling, knowledge management and distributed systems development. He was project leader in several Spanish organizations. He has a B.S. and Ph.D. in computer science. He is author of several book chapters and publications in software engineering.

Sonia Suárez. Mrs. Suárez received the Computing Engineer degree from the University of A Coruña, Spain. Actually, she is a PhD. Student at the Polytechnical University of Madrid, Spain. Her research interests in computer science include knowledge management, e-learning, validation and experimentation. She is author of several publications in software engineering.