

# **A Context-Based Knowledge Management Framework for Software Development**

*Ching Kang Cheng, Franz J. Kurfess,*  
California Polytechnic State University, San Luis Obispo, CA, U.S.A.  
ckcheng@calpoly.edu, fkurfess@calpoly.edu

## **Abstract**

Verifying the match between the expectations or requirements of a customer against the actual functionality of the implemented system is very critical for the success of a system, but may also be very difficult to achieve. The basic problem is often to capture the intended meaning of statements in documents that describe what the customer wants, translate it into a format suitable for computers (such as a formal specification), and compare it with the functions that the implemented system performs. In practice, this usually relies heavily on the experience and familiarity of the developer both with the domain as well as the system implementation. In this paper, we describe a context-based knowledge management framework that quickly identifies and retrieves relevant documents, and helps with the construction of an ontology reflecting the respective domain knowledge. With this framework, the gap between customer expectations and system implementation can be narrowed, thus enhancing the testing and quality assurance process.

## **Keywords:**

knowledge management, requirements engineering, ontology, context, software quality assurance, software quality management

## **1.0 Introduction: Why context-based knowledge management?**

The software quality management and assurance process is highly complex. It requires various skills and knowledge such as QA techniques, software developing skills and domain knowledge [1]. Much of this required knowledge is encapsulated in unstructured or semi-structured formats such as software requirement documents, software design documents, software external interface documents, technical journals, bugs reports and user reviews. In software engineering, knowledge is also highly contextual in nature. For example, a solution to a connection error message for one database does not apply to another database. Therefore, to capture the real essence of a piece of knowledge, it has to be captured within its context [2].

In this paper, we present a framework where knowledge can be extracted and categorized with reference to some specific context known as ontology. It is possible for this ontology to be extended dynamically. In this way, new knowledge can be captured and categorized according to its context. In addition, the resulting knowledge base can be queried using the ontology as the search index. We adopt the context-based approach enabled through the use of an ontology in our proposed framework. In section 2, the various processes and activities in the knowledge management framework are described. Section 3 discusses the system architecture of the framework and the different components involved. In section 4, we summarize our findings and outline future endeavors.

## 2.0 Context-based knowledge management activities in software assurance

We argue that knowledge has to be organized in a fashion that is intuitive for the user to be useful. We propose the context-based knowledge management (CKM) framework which allows knowledge to be categorized semi-automatically in a manner consistent with the user context. Figure 1.0 illustrates the overall process of categorizing knowledge according to the user perspective.

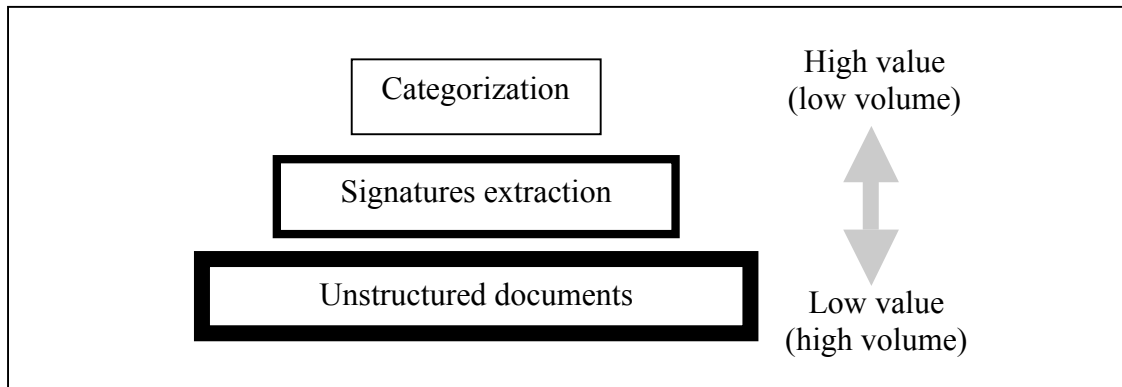


Figure 1.0: The overall knowledge management process

### 2.1 Ontology

An ontology provides an explicit formal specification for the terms used in a particular domain, and identifies relations among these terms [6]. Such an ontology can be used as a common framework among various parties interested in the domain{ XE "domain" }. It identifies and formalizes the underlying structure of the information and knowledge about the domain. An ontology is basically a graph whose nodes represent the concepts or objects of a domain, and the edges indicate relationships between concepts. Usually this graph is structured around a hierarchical “backbone” similar to the class/subclass relationship in object-oriented programming. It is not a strict tree, however, and links can exist between any concepts in the graph. Due to the formalization, it can be represented and to some degree interpreted by machines, and enables the formal analysis of the domain. This allows an automated or computer-aided extraction and aggregation of knowledge from different sources and possibly in different formats (as long as the formats can be mapped to the ontology).

To a certain extent, ontologies can mirror class hierarchies, objects, relation, properties, and methods used in software development. The latter, however, usually reflect the perspective of software developers, whereas ontologies concentrate on aspects of the domain that are visible to all parties interested in a particular domain, in particular users of software applications. From an engineering perspective, ontologies can be very helpful with the reuse of domain knowledge, and for the separation of domain knowledge and software code that performs operations on that knowledge. In the CKM framework, an ontology is used in both the signature extraction engine and the categorization engine.

## 2.1 Unstructured documents

In the software quality and assurance domain, valuable knowledge is encapsulated in unstructured or semi-structured documents such as software requirement documents, software design documents, bug reports, technical journals and user reviews etc. Another source for knowledge is the World Wide Web and the semi-structured documents referred to as Web pages.

When a quality agent (quality manager or test engineer) [1] finds a bug, say, “ODBC: 1234” while testing, he or she can look into bug reports or technical journal and read about how one’s peer solved the problem when encountered. This prevents reinvention of the wheel and makes debugging a smoother experience. Furthermore, to keep pace with the latest development in software technology, the quality agent has to constantly extract knowledge from this large collection of information. Undoubtedly, these unstructured or semi-structured documents serve as an important source of knowledge in the software assurance domain.

## 2.2 Signature extraction

In CKM framework, a signature refers to a set of keywords that links a specific document to a concept within the ontology. For example, the signature *TRANSPORT* may contain the keywords car, train, plane and ship, and the signature *CAR* the keywords BMW, Ford, Daimler-Chrysler, police vehicle and taxi. The ontology enables the computer (CKM) to correlate the occurrence of the words in the unstructured documents with the respective meaning in the user context.

The objective of the signature extraction engine is to construct meaningful signatures according to the ontology from the unstructured documents. Each time a signature is found in an unstructured document, an instance of the signature is created. For example, if an unstructured document contains twenty occurrences of the word Honda, twenty instances of the signature *CAR* will be instantiated and stored in the signature knowledge base. The signature knowledge base forms the input for the categorization engine.

## 2.3 Categorization

A category is defined as a grouping of signatures that have the same perspective. For example, the signatures *TRANSPORT* and *CAR* are grouped in the category *HOBBY*. An ontology defines the categories; each category corresponds to a node in the ontology. One benefit is that it is possible to extend the ontology dynamically. For example, a new category *INVENTION* containing the signatures *TRANSPORT* and *CAR* can be added to the ontology in run time, without re-compiling the source code of the categorization engine.

The categorization engine will group the extracted signatures into categories defined in the ontology. Each category object contains information about the physical location of the unstructured document and the relevance of this document with respect to the

category. For example, if an unstructured document contains twenty instances of the signature *CAR*, a new category instance *HOBBY* will be created. In addition, the *HOBBY* category contains a property weight which is set to twenty to reflect the number of occurrences of the signature. Also, the property file-location is set to the physical location of the unstructured document. This new category is added to the category knowledge base.

## **2.4 Accurate and fast retrieval of relevant knowledge**

It is critical that the relevant knowledge can be identified and retrieved quickly. For example, when a quality agent encounters a technical problem, the learning curve is shortened if the solution can be reached easily and quickly. Another example is the derivation of test plan and test procedures. Again, the overall effort and time taken for this process can be shortened if the required knowledge can be retrieved efficiently. The design of the CKM framework enables knowledge to be classified in a user-defined context. Hence, we believe that such an approach can fasten the retrieval of relevant knowledge and ultimately aid the quality agents to perform their tasks proficiently.

## **2.5 Ability to extend the knowledge category dynamically**

As the software development evolves, new knowledge arises. Inevitably, the process of knowledge extraction and categorization will iterate. For example, a quality agent discovers a new method for solving an existing problem. The existing category must be extended dynamically to include this new knowledge. The CKM framework allows the user to create a new category dynamically through a graphical user interface. More importantly, this is achieved without re-compilation of the source code. In this way, the categories are extended according to the user context and are synchronized with the software development. While our framework supports the automatic extension of an ontology, this is not implemented in the current version.

### 3.0 System architecture

The core design principal of the CKM framework is to provide loosely coupled yet seamlessly integrated components. To achieve this, the CKM framework architecture is decomposed into three distinct layers and the interfaces between the components are specified in a language neutral format (e.g. via XML), as shown in Figure 2.0.

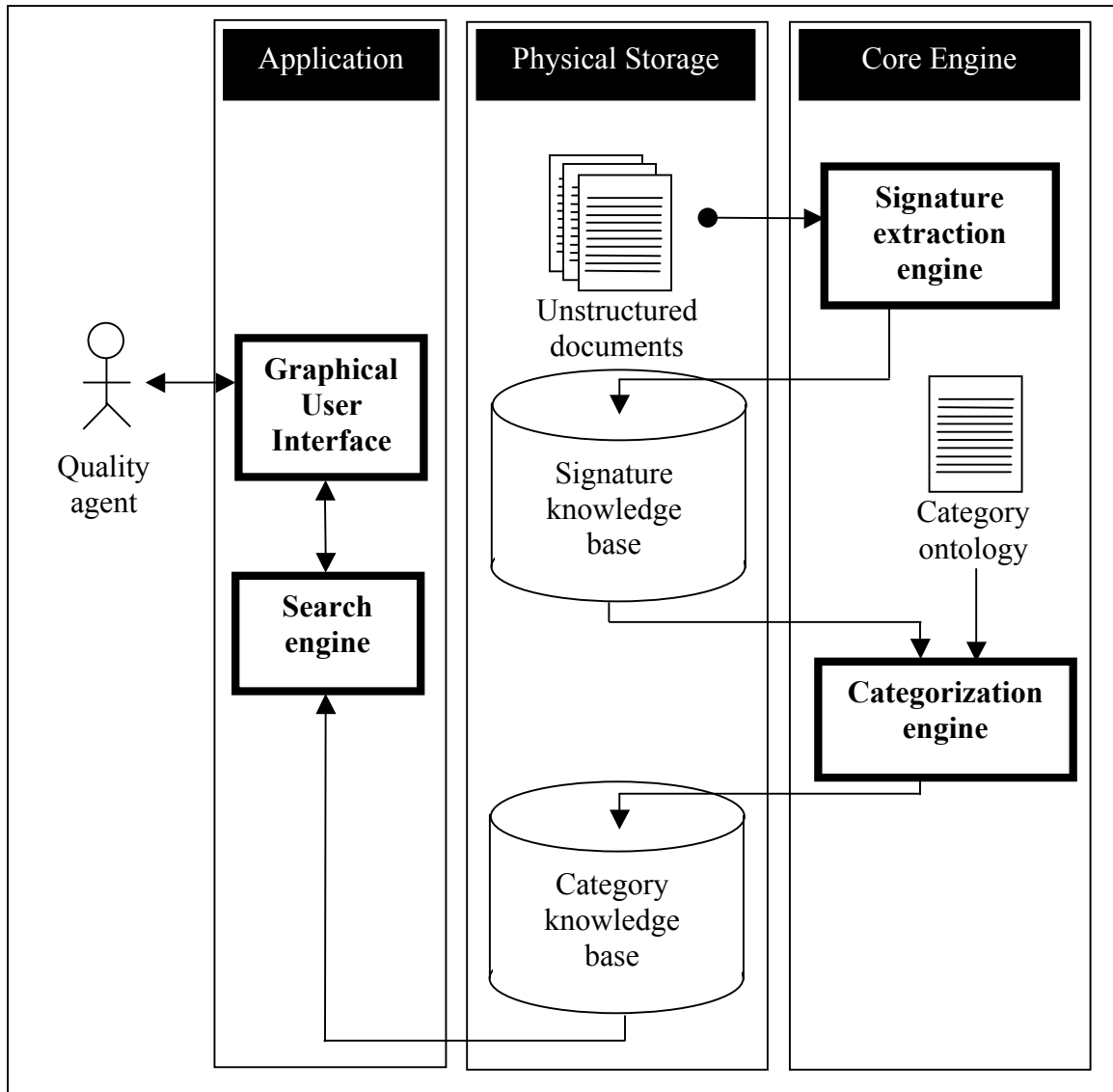


Figure 2.0: System architecture of the CKM framework

#### 3.1 Core engine layer

The core engine layer encompasses components that contribute to the core functionality of the framework. It contains the signature extraction engine and the categorization engine.

### 3.1.1 Signature extraction engine (SEE)

The signature extraction engine has been developed at California Polytechnic State University San Luis Obispo [5]. It provides the functionality to extract the meaning of a sentence through the application of a context model and subsequently represents the meaning of the sentence in the model [5]. The three main components are: Link Grammar Parser, WordNet Database and Mapping Engine.

#### 3.1.1.1 Link Grammar Parser

The Link Grammar Parser is a syntactic parser of English, based on link grammars, an original theory of English syntax [3]. The Link Grammar Parser has been integrated with SEE to assign a syntactic structure to a given sentence, which consists of a set of labeled links connecting pairs of words.

#### 3.1.1.2 WordNet

WordNet is an online lexical database designed for use under program control. English nouns, verbs, adjectives, and adverbs are organized into sets of synonyms, each representing a lexicalized concept. Semantic relations link the synonym sets [4]. In SEE, lexical knowledge is acquired through integration of the system with the WordNet database, and contextual knowledge is acquired by tracking contextual meanings of words and phrases during and after development of ontology (i.e., context model).

#### 3.1.1.3 Mapping Engine

The Mapping Engine performs the matching of the sentence from natural language to a context model. Different context models may produce different results simply because words could have different meanings in different contexts. In SEE, the representation of meaning is accomplished by manipulations of a context model (i.e., creation, modification, and deletion of objects and relationships in an object model). The relationship between the components above mentioned is shown in Figure 3.0 [5].

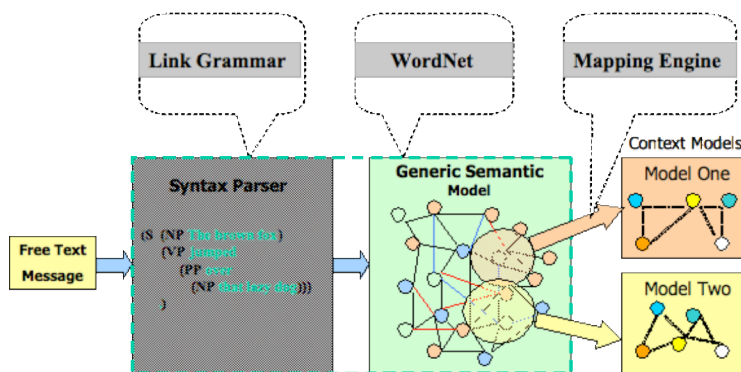


Figure 3.0: SEE system architecture [5]

### 3.1.2 Categorization engine

A key feature of the categorization engine is that the category ontology can be extended dynamically, i.e. it is not constrained by any particular knowledge domain. A system change from one ontological model to another does not require significant system reconfigurations. For example, a new category “CORE DUMP” can be created and added to the category ontology through Protégé-2000. The categorization engine is able to include this new change without any reconfiguration.

Again, with the component-based approach, the interfaces of the categorization engine are designed to be language neutral. The inputs are from the signature instances (signature knowledge base) and the category ontology. The outputs are the category instances (category knowledge base). Figure 4.0 illustrates the interfaces of the categorization engine.

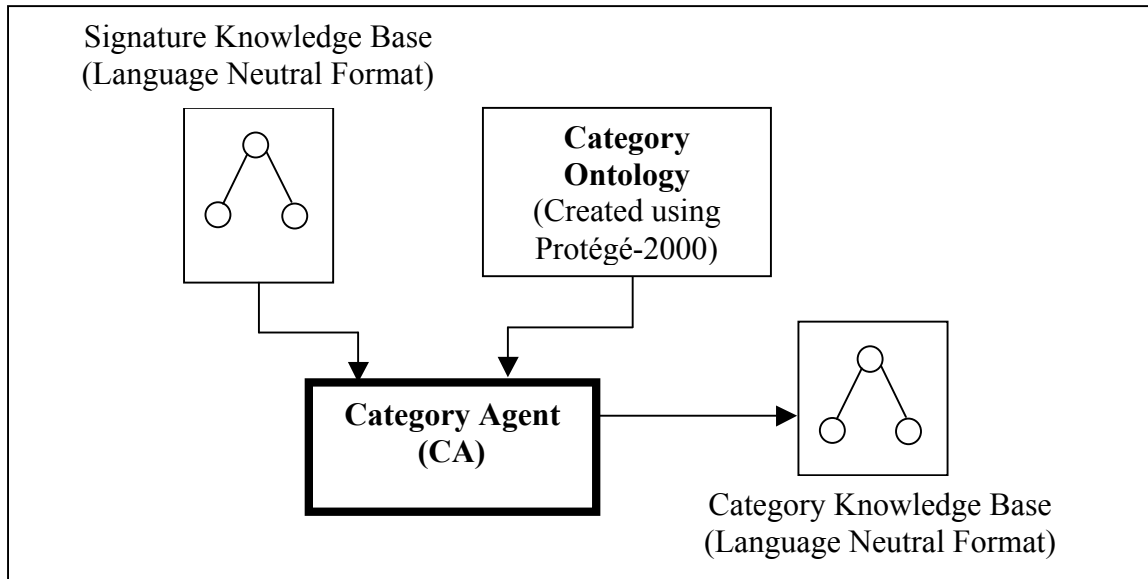


Figure 4.0: Interfaces of the Categorization Engine

#### 3.1.2.1 Protégé-2000

Protégé-2000 is an extensible, platform-independent environment that allows developers to create and edit ontologies and knowledge bases. Protégé-2000 provides an ontology editor that both developers and domain specialists can use. The user interface allows multiple views of ontologies and knowledge bases. Through the utilization of tabs, users can switch between these views [7].

Protégé-2000 is used as the ontology editor for the user to update the ontologies (context model). At the same time, it serves as a viewer where the user can visualize the relationships among the categories and the content of the knowledge base. For example, the user can determine what are the documents categorized under a particular category by navigating through the hierarchy structure of the categories.

## 3.2 Physical storage layer

The physical storage layer handles the storing of the knowledge base and the documents. The interface between the physical layer and the rest of the components is defined a language neutral format such as XML, ensuring a loose coupling between the different layers. Applications that have to interact with the physical layer can be written in any programming language as long as that language supports XML. On the other hand, the knowledge base and the unstructured documents can be stored in text file format, binary file format, relational databases or object-oriented databases.

## 3.3 Application layer

The application layer is a logical grouping of components that leverage on the category knowledge base. Examples are search engine and research assistants. By design of the CKM framework, application components can be plugged into the framework as and when they are ready.

### 3.3.1 Search Engine

The search engine allows the users to query the category knowledge base using the context ontology as search criteria. We believe that the search is more accurate and helpful when the users can relate the search category with the domain knowledge. This is possible as the search category is created and maintained with the user's participation.

Since the category knowledge base is implemented in a neutral language format, the search engine can be implemented with various programming languages and toolkits. Protégé-2000 provides a query interface which can be used for this purpose. However, this query interface does not cater for any advanced search functionality. Figure 5.0 illustrates the interface of the search engine.

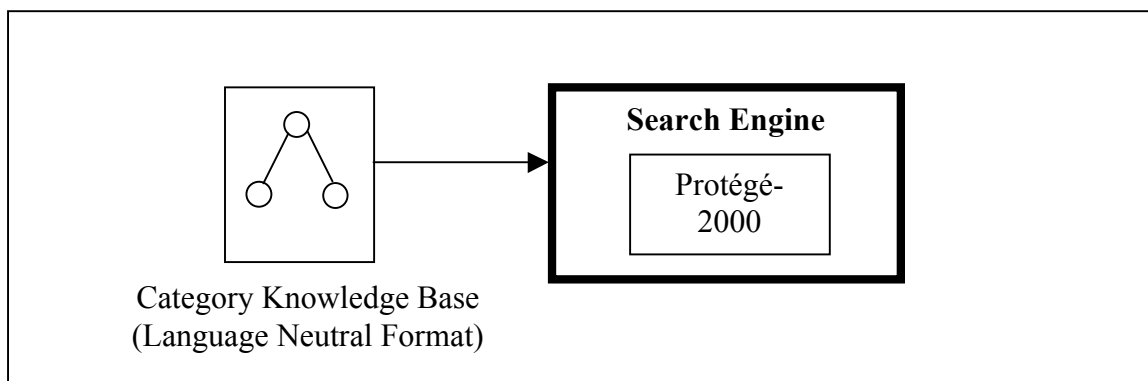


Figure 5.0: Interfaces of the Search Engine

## 4.0 Conclusions and future work

The purpose of the Context-Based Knowledge Management (CKM) framework is to explore the opportunities and value of a context-based approach in a domain where knowledge is encapsulated in massive collections of unstructured documents. We believe that the software quality process in a large software project is an environment that benefits from the context-based approach we have presented.

An enhancement to the Signature Extraction Engine (SEE) is to extend the usage of the WordNet database, which provides the SEE with lexical knowledge about the English language [5]. The current implementation includes only two types of relationships: synonymy and irregular forms of words. Inclusion of additional types of relationships present in WordNet can enhance SEE's capabilities to deal with natural language.

We will continue to develop applications that leverage on the category knowledge base. One such application is a research assistant whose main functionality is to automatically provide additional information relevant to the context of the research that the user is working on. We hope that such applications supported by the CKM framework can convert information stored in unstructured documents into valuable knowledge that can be retrieved easily and accurately.

## 5.0 References

- [1] Kurt Schneider  
Realistic and unrealistic expectations about experience exploitation  
Conference on quality engineering in software technology Conquest 2001, 19-21 September 2001
- [2] Kevin C. Desouza  
Technical opinion: Barriers to effective use of knowledge management systems in software engineering  
Communications of the ACM January 2003  
Volume 46 Issue 1
- [3] Temperley, D., Sleator, D., and Lafferty, J.  
An Introduction to the Link Grammar Parser  
Technical report, Available: <http://www.link.cs.cmu.edu/link/>  
March 1999.
- [4] George A. Miller  
WordNet: A Lexical Database for English  
Communications of the ACM November 1995  
Volume 38, No. 11
- [5] Xiaoshan Pan  
A context-based free text interpreter  
California Polytechnic State University San Luis Obispo Master's Thesis - Computer Science Department Aug 2002
- [6] T. R. Gruber  
A Translation Approach to Portable Ontology Specification  
Knowledge Acquisition 5: 199-220, 1993.
- [7] M. A. Musen, R. W. Ferguson, W. E. Grosso, N. F. Noy, M. Crubezy, & J. H. Gennari.  
Component-Based Support for Building Knowledge-Acquisition Systems  
Conference on Intelligent Information Processing (IIP 2000) of the International Federation for Information Processing World Computer Congress (WCC 2000), Beijing 2000