Constructive Knowledge Management Model and Information Retrieval Methods for Software Engineering

Zeyar Aung[1] and Khine Khine Nyunt[2]

[1]Computing and Information Science Program, Masdar Institute of Science and Technology, United Arab Emirates. Email: zaung@masdar.ac.ae

[2]College of Computer Science and Information Technology, King Faisal University, Kingdom of Saudi Arabia. Email: knyunt@kfu.edu.sa

Abstract:

In this book chapter, we will discuss the two important trends in modern software engineering (SE) regarding the utilization of knowledge management (KM) and information retrieval (IR). Software engineering is a discipline in which knowledge and experience, acquired in the course of many years, play a fundamental role. For software development organizations, the main assets are not manufacturing plants, buildings, and machines, but the knowledge held by their employees. Software engineering has long recognized the need for managing knowledge and that the SE community could learn much from the KM community. We will introduce the fundamental concepts of KM theory and practice and mainly discusses the aspects of knowledge management that are valuable to software development organizations and how can a KM system for such an organization be implemented. In addition to knowledge management, information retrieval (IR) also plays a crucial role in SE. IR is a study of how to efficiently and effectively retrieve a required piece of information from a large corpus of storage entities like documents. As software development organizations grow larger and have to deal with larger numbers (probably millions) of documents of various types, IR becomes an essential tool for retrieving any price of information that a software developer wants within a short time. IR can be used both as a general-purpose tool to improve the productivity of developers or as an enabler tool to facilitate a KM system.

1. Knowledge Management

Knowledge management is fundamentally corporate intellectual assets to improve the organization's effectiveness, as well as its business opportunity enhancement. Key to knowledge management is capturing tacit knowledge for the tangible benefits for the organization. The aim of knowledge management is to continuously improve an organization's performance in which sharing, creating, assimilating, disseminating, and applying knowledge throughout the organization. Knowledge management is a continuous process to understand the organization's knowledge needs, the location of the knowledge, and how to improve the knowledge.

1.1 Fundamental Concept of Theory and Practice

Knowledge is one of the organization's most important value and influencing its competitiveness. In this age of information organizations see their people as their key assets. The knowledge, skills and competencies of these people add to the growth of the organization.

The first section of this topic presents fundamental knowledge – the "tacit" or personal knowledge versus the "explicit" or organizational knowledge (Nonaka and Takeuchi, 1995). Tacit knowledge resides in individuals and teams which includes personal experiences, thinking, competence, perceptions, insights and know-how that are indicated but not actually expressed. Explicit knowledge that is codified and conveyed to other thought which will be transformed to data, information later documents, records and files.

As Nonaka and Takeuchi (1995) illustrate that there are four ways to transform the knowledge. Firstly, "Socialization" means to share experience from tacit knowledge to tacit knowledge. This process is first to share experience and then to exchange tacit knowledge. Thus, socialization is used in sharing learners' experience and know-how with other learners. The second concept is "externalization" that means the conversion of tacit knowledge into explicit knowledge. This process is to rationalize tacit knowledge and articulate it into explicit concept. Third one is "Internalization" that is a process of

embodying explicit knowledge into tacit knowledge. Individual gained knowledge and experience through the explicit knowledge and individual can develop the new tacit knowledge internally. The fourth one is “Combination” that converts explicit knowledge into more complex and systematic sets of explicit knowledge. In this process, individuals combine and exchange different explicit knowledge to explicit knowledge with others.

The second section presents the Five Learning Cycles model of “organizational learning” (Sanchez, 2001). In this general model of learning processes in an organization, five kinds of learning cycles are identified that link of individuals, groups, and the overall organization in an organizational learning process.

Sanchez (2001) illustrates that the first learning cycle is “Individual Learning Cycle” whose individuals imagine alternative interpretive frame works and new kinds of knowledge. The second learning cycle is “Individual/Group Learning Cycle” who shares their new knowledge within groups to evaluate the new knowledge developed by individuals. The third learning cycle is “Group Learning Cycle” whose interact with other groups to determine whether new knowledge developed by a given group becomes accepted within the overall organization. In this stage managers or leaders are the domain expert of the learning. The fourth learning cycle is “Group / Organization Learning Cycle” in which new knowledge accepted at the organizational level which is embedded in new processes, systems, and the culture of an organization. The fifth learning cycle is “Organization Learning Cycle” which is new knowledge embedded in new processes, systems, and organizational culture which leads to new patterns of action by groups and individuals.

As Sanchez (2001) illustrate about basic assumptions in personal versus organizational knowledge management. Firstly, Personal knowledge in nature is very difficult to extract. In contrast, organizational knowledge can be articulated and codified to create as an organizational asset. As a result of this stage individuals knowledge can be articulated and transformed to explicit knowledge which can be references for others individuals. Secondly, personal knowledge must be transferred by people to people. In contrast, organizational knowledge can be disseminated through Information

Technology to be transformed as an explicit knowledge such as documents, records and files.

Thirdly, Learning can be encouraged by bringing the right people in the right time in the personal knowledge. On the other hand, learning process can be created through definable and manageable knowledge explicitly throughout the organization. As a result, Organizational knowledge approach focus on designing organizational knowledge processes for generating, articulating, categorizing and leveraging throughout the organization.

1.2 Knowledge Management in Software Engineering

Software engineering is a knowledge intensive business and as such it could benefit from the ideas of knowledge management. The important question here is: where does knowledge resides in software engineering? Software engineering involves a multitude of knowledge-intensive tasks: analyzing user requirements for new software systems, identifying and applying best software development practices, collecting experience about software development process, project planning and risk management, and many others (Birk et. al., 1999).

In this section, we will present the perspective of KM for SE such as knowledge management support for core activities in SE, organizational memories for software development and classification of knowledge management tools relevant to SE.

1.2.1 Knowledge Management Support for Core Activity in Software Engineering

This section addresses major core KM activities in software engineering processes such as Document Management and Competence Management. Document Management system allows for the storing and uploading files, search and retrieval and the search for experts based on content. On the other hand, Competence Management represents the management of tacit knowledge is vital to the organization through as explicit knowledge.

Document Management: Birk et al. (1999) illustrate the wide spectrum of software engineering processes that might occur in a typical software engineering project. A software development project involves a variety of document-driven processes and activities. Document Management is mainly focus on authoring, reviewing, editing, and using these documents which becomes the main sharing knowledge system throughout the organization. There are many tools to support DM system such as Hyperwave, Microsoft Sharepoint, Lotus Domino, and Xerox DocuShare which include features such as defining process workflows and finding experts. Therefore, document management is a basic activity toward supporting an organization's implementation of a knowledge management system.

Competence Management: CM is more difficult to identify and difficult to manage tacit knowledge that resides in individual experts and team. It means that not all tacit knowledge can be transformed explicit knowledge. So, an organization must track who knows what to fully utilize undocumented knowledge. This process becomes the transformation of tacit knowledge to explicit knowledge which becomes the organization's assets. In general, we should form the small group to capture knowledge in fact larger group of people are exposed to the risk of “not knowing what other people know.” An elaborated solution to this problem is competence management, skills management or expert network.

Initially, the main objective of Competence Management systems is to find individuals who have specific pieces of knowledge but later use to generate and edit their own profile by using CM tools such as Skillscape and Skillsoft.

1.2.2 Organizational Memory for Software Development

Organizational memory is both an individual- and organizational level construct (Walsh and Ungson, 1991). However, Individual memory is not sufficient and the entire organization needs a memory to explicitly record critical events. Software Engineers conduct their daily work often supports to create such a memory in the SE environment. There are many SE practices to build such memories such as version control, change

management, documenting design decisions, and requirements traceability which will effect directly or indirectly to software development.

Version control system also known as Source Code Control System, represent a class of tools that indirectly create a project memory. The system recoded the information about who made the change, why need to be change and when they change. This memory helps the software's evolution after the project. Then domain expert has to identify to use this information for advanced analysis of software products and processes.

Design Rationale is an approach to capture the software design decision explicitly to avoid repeating mistake as well as create a product memory. This memory will definitely help engineers to test different technical solutions and make decision during the design process. But how engineers make decisions which are rarely captured because it difficult to understand the reason behind the solution.

Software requirements drive the development of software systems, but the connection between the final system and its requirements is fuzzy.

Traceability is an approach that explicitly connects the requirements and the final software system. This memory will help what type of requirements led to a particular of source code, what type of code did engineers develop to satisfy the particular requirement.

1.2.3 Classification of Knowledge Management Tools

In this section, we present types of commercial and academic KM tools. Most of the commercial tools are search and database maintenance, intranet features, FAQ lists, logged chat features, find-an-expert features, personalization, etc., which aid in knowledge-sharing within an organization. Some of the tools are Experience Management System, Case-Based Reasoning (CBR) for retaining and retrieving experience.

The tool named BORE (Building and organizational Repository of Experiences) was developed by the University of Nebraska-Lincoln. BORE is case studies based tool which is related to real time problems solving experiences so that Software developers can use these solutions in future projects.

Mostly IT based tools that can help to fulfill KM goal. IT based tools are categorized as bellows:

Groupware Systems: It is a supporting tool to determine the processes that take place in the organization as well as how knowledge is currently stored and distributed, and establish how certain functions would improve them. publishing and communication tools, collaborative management tools, video conferencing and informal communication tools provides facilities to create, share explicit knowledge as well to find the sources of the knowledge. One of the best examples of groupware tool is Lotus Notes.

Decision Support Systems: The role of this system is to access and manipulate data based on Data Warehouse and Data Mining technology as well as online analytical processing system (OLAP). The goal is to enhance decision –making and solving problem.

Content Management Systems: CMS is responsible for the creation, management, and distribution of content on the intranet, extranet, or a website which is relevant to knowledge management (KM). Although such systems deal almost exclusively with explicit knowledge, the sheer volume of documents that an organization has to deal with makes them useful and in some cases even mandatory

1.3 Constructive Knowledge Management Model

This section presents our proposed “constrictive knowledge management” model that entails different methods/forms of KM for different steps of software engineering. The model makes clear how new knowledge developed by individuals in a software development organization must navigate each of the Five Learning Cycles (Sanchez, 2001) to become accepted by other people in the organization, and then how new

knowledge becomes embedded in the organization and its way of working. In effect, the model shows at the macro level how personal knowledge is converted into organizational knowledge, and vice versa, in processes for active and continuous organizational learning.

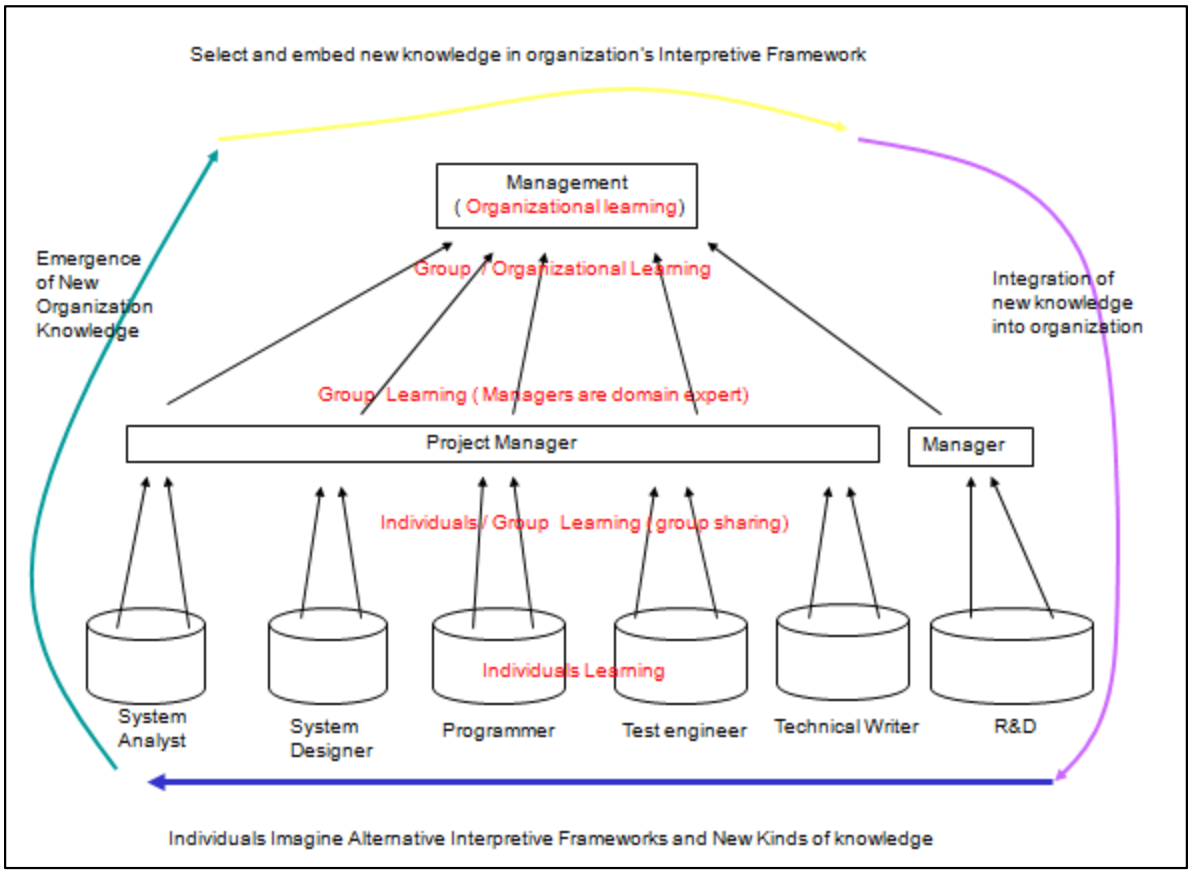

Figure 1: Constructive Knowledge Management Model.

We start by showing in Figure 1, we understand by "organization" a company or a business unit within a company whose core business is to develop software and which carries out activities to improve its software practices and processes. we consider a software development project team which include Project Manager who lead the team, System Analyst - a specialist who studies the problems and needs of an organization to determine how people, data, processes, and information technology can best accomplish improvements for the business, System Designers - a technical specialist who translates system users' business requirements and constraints into technical

solution, Developers who implements the software components, Test Engineers who performs unit and software integration testing, Technical Writers who produces and maintains the software design document and associated models, and R&D team for the post project evaluation. Our model is based on the five learning cycle to generate, disseminate and apply knowledge within the organization.

In the first cycle, individuals generate imaginary alternative interpretive frame works and new kinds of knowledge which is tacit knowledge. It can be any best practice of software development or new method of solving problem or new approach of software methodology, etc. In the second cycle, organization would have workshop or discussion for all individuals who share their new knowledge within groups to evaluate the new knowledge developed by individuals. In this section, group member would have open discussion about new idea can be accepted or rejected. In the third cycle, groups who interact with other groups to determine whether new knowledge developed by a given group becomes accepted within the overall organization. In this stage the domain experts (manager or leader) are the main player to emergence the new knowledge to the organization. The fourth cycle, group or organization have to select new knowledge accepted at the organizational level which is embedded in new processes, systems, and the culture of an organization's interactive framework. The fifth cycle, new knowledge is embedded in new processes, systems and organizational culture which lead to new patterns of action by groups and individuals. Finally, new knowledge would be integrated into the organization.

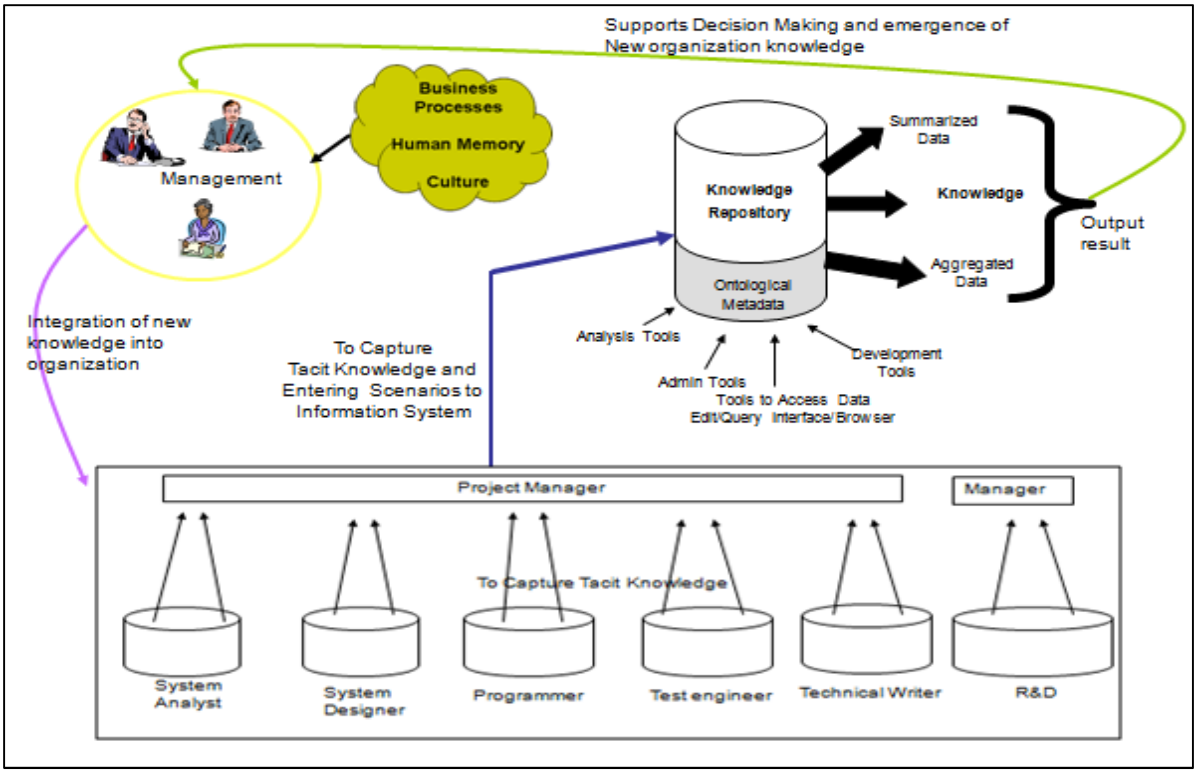

Figure 2: General Overview of Constructive Knowledge Management Model.

Then we introduced Figure 2, the detail model of how knowledge was articulate to the organization through Information Technology system and how IT system is being used for this regards.

After capturing the tacit knowledge from the individual, group or organization learning cycle was introduced to evaluate and defined the new knowledge which is approved by domain expert, mostly are managers and leaders. Then domain expert is responsible to enter scenarios to the information system by using standard format which is approved by organization. After storing new knowledge in the repository, we can generate the best practice and new organization knowledge by summarizing data, aggregate data by using various analyzing tools and query on ontological meta data which is a back bone for providing and accessing knowledge sources. The word “ontology” is a terminology for the knowledge indexing and searching process. When you do comparison for key-word based indexing or searching, an ontology will be formalized, common and shared the description of a domain which is the main advantages of indexing and searching

process. Then the output result will support for decision making and emergence of new organizational knowledge which must be integrate with business process and organizational culture.

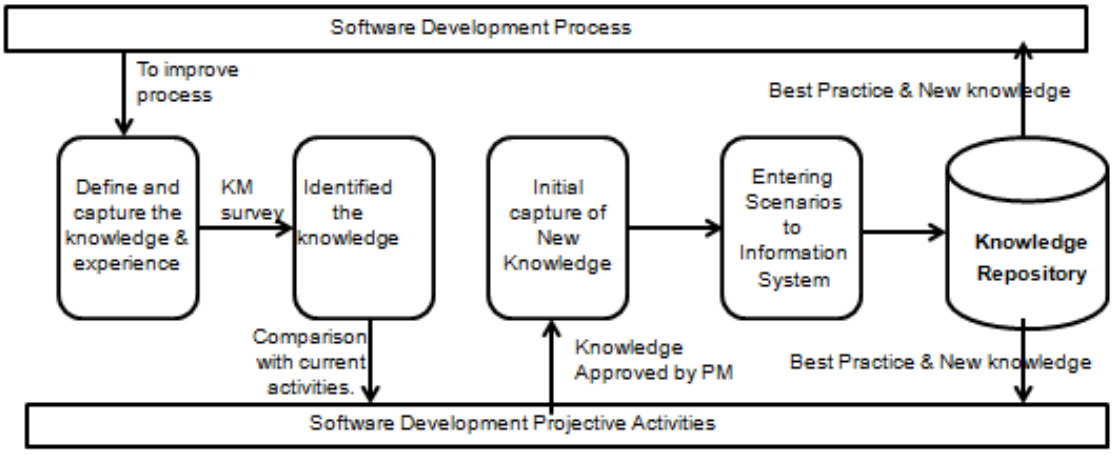

Figure 3: Detailed Process of Knowledge Capturing.

Figure 3 shows the detail process of the main activities of our proposed model, from the initial definition of the objectives of knowledge and experiences that are intended to capture, until its end, when the new knowledge and best practices are identified and integrated to the software development process as well as project activities.

Firstly, organization defines a series of objectives for new knowledge creation for those software development practices and processes that the organization considers necessary to improve. Based on these objectives the “KM Survey” is developed, defined as a knowledge management which purpose is to capture new knowledge and experiences. The survey contains a series of questions or multiple choices so that individuals have an opportunity to share of the knowledge and experience in the KM workshop or particular discussion time which is officially set by organization.

Secondly, after survey has done, domain expert (manager or leader) has to verify the outstanding answers given by individuals. This is very initial stage to identify the new knowledge. This new knowledge should go for the group discussion to get opinions for

other individuals whether accepted or rejected. Then domain expert are the main key person to do comparison with current practice and decide to adopt as a new knowledge to improve the software development process or project activities.

Thirdly, once these survey are elaborated, individuals are assigned to execute software development process or project tasks for a primary analysis for the initial identification of new knowledge and experiences captured in the answer which is to register their reflections and impressions, difficulties found, unforeseen facts and similar considerations related to the manner in which they carry out his/her tasks. At this stage, domain expert has responsible to monitor the outcomes.

Lastly, new knowledge and personal experiences will be incorporated into the Repository and these knowledge and experience will impact on how software process and project activities will be carried out in the future, and be the basis for future improvements to the software practices and processes used by the organization.

This sequence of activities is repeated iteratively in order to enable the organization to manage, in an incremental manner, the creation of knowledge. In this way, it is possible to gather more insights about their appropriateness and to allow the organization to continuously refine the knowledge and experience already captured.

2. Information Retrieval

Information retrieval (IR) is a study of how to effectively and efficiently retrieve a piece of information that a user wants from any form of large-volume storage like a collection of files, documents, databases, knowledge bases, or ultimately the World Wide Web. Internet search engines like Google are popular examples of information retrieval tools. Large software development organizations has to deal with thousands or even millions of documents of a variety of types for every step of the software development life cycle — like feasibility reports, various design diagrams and reports, program source codes and comments, test scripts and reports, maintenance reports, etc. — for each of its software projects. (There may be hundreds of such projects in a big software house.) Thus, there is a pressing need for a tool to organize all of these documents in a

systematic way and enable the retrieval of a piece of information required by a software developer in an easy and efficient manner. A dedicated IR system for software engineering will be able to address these requirements and will help make the life of a developer easier. In fact, an IR system can be used both as a generic tool for retrieving any pieces of information that developers needs from a large corpus of documents in the development life cycle of a software project or as a facilitator tool to support a software engineering KM system (e.g., in retrieving a piece of encoded knowledge from a knowledge base).

2.1 Methods and Tools in Information Retrieval

In this section, we will present the methods and tools that are available in IR — especially those relevant to the SE business. We will explore the distinct yet interrelated IR techniques of corpus building, indexing, query evaluation, and evaluation of retrieval effectiveness.

Given a collection of documents, we have to transform them into a specific representational form, called a corpus, to facilitate easier handling, and then build an index on it for faster retrieval.

2.1.1 Corpus Building

Since the original documents are normally in free formats, like plain text files, pdf files, word processing (like MS Word) files, XML files, or program source code files, extracting useful textual information form them is an important exercise. Various file parsers and text segmentation tools (Choi, 2000), depending on the source file type, could be used for information extraction purpose. After the extracting textual information, we have to preprocess it in order to make it cleaner and more standardized.

In the preprocessing step, we usually convert a given text into a “bag of words”. That means tokenizing (i.e., splitting) of the text strings into a set of bare words by removing punctuation marks. Then, non-informative stop words such as articles, conjunctions, and prepositions (for example, “a”, “the”, “who”, “from”, “this”) are removed. After that,

we carry out stemming, in which all the derived words are converted into their corresponding base words. For example, if the base word is “fish”, the derived words are “fishing”, “fished", "fish”, and “fisher”, etc.

After extracting the textual information from the original documents and preprocessing them, we build the resultant texts into a corpus. This involves storing them in a standardized “internal” format, which can be in the form of plain text files, tables in relational databases, or distributed file systems like MapReduce or Hadoop. These new files/records can be regarded as cleansed versions of their original documents. Each file/record in the corpus can be linked to its original document for back referencing in the future. The purpose of building the corpus is to enable easier manipulation of all the pieces of information in the IR system.

2.1.2 Indexing

After building a corpus, we usually construct an index on it with a view to faster retrieval. This step is essential when the number of documents in the corpus is large, usually containing thousands or even millions of documents, as in the case of a corpus for a big software house. An index can take one of many possible forms. One of the classical examples for document indexing is **inverted file indexing** (Baeza-Yates and Ribeiro-Neto, 2011). We store a main list of distinct words, and each word in that main list points to individual “posting lists” containing the IDs for the documents in which it occurs, its frequency in each document, and optionally, its positions of occurrences. Another type of document indexing is a **signature file**. Given a list of words of interest, a signature for each document is a bit vector which encodes the presence (1) or absence (0) of each word in it. Usually, a hash coded version of that signature is stored in order to reduce the storage space. This hash-coded signature will serve as a mask to filter out many irrelevant documents during the query processing.

Another popular model for indexing and processing of documents is called a **vector space model** (Baeza-Yates and Ribeiro-Neto, 2011). Each document is regarded as a single data point in a high dimensional space (i.e., a data vector with a high number of

attributes.) Each dimension in this model represents a distinct word in the corpus. That is, if there are t unique words in our corpus, each document is a t-dimensional data point (vector). In its simplest form, the attribute value for each dimension in a document vector is 1 if the corresponding word is present in the document and 0 if the word is not. In practice, more sophisticated mechanisms such as normalized word weights are used.

Since t can be a very large number for large corpuses, the techniques of dimensionality reduction (such as Principal Component Analysis) or unsupervised feature selection (like random feature projection) may be used to reduce the number of dimensions of a document vector. This is done in order to avoid the "curse of dimensionality", a phenomenon which negatively affects both speed and accuracy in query processing. Indexing techniques can be used to index the data points either in their original or reduced dimensional spaces. High dimensional indexing techniques like iDistance, kd-tree, and X-tree or filtering techniques VA-File can be used (Samet, 2006).

An enhance version of the vector pace model is known as the **latent semantic indexing** model (Deerwester et al., 1990). The vector space model assumes independence among words in a document. However, this does not reflect the reality. The latent semantic indexing utilizes the rich expressive power of natural language like English. It groups semantically related words into "concepts" using singular value decomposition. The resultant concepts are regarded as independent of each other.

2.1.3 Query Evaluation

The simplest form of queries for document retrieval is a **Boolean query**. A few key words and the Boolean operators between them are used to retrieve the documents that contain those words. Such queries are equivalent to normal database searches. For example, a query like "`renewable` **`AND`** `(power` **`OR`** `energy)` **`AND`** `(solar` **`OR`** `photovoltaic` **`OR`** `PV)`" will retrieve documents that contain information regarding renewable energy generation using solar panels.

For the **rank queries**, the documents in the corpus are ranked according to their putative relevance to the query key words (like Google does when we search our key

words). The putative relevance is measured using the concept of "similarity" between the documents. "Distance" is the flip concept of similarity. We can merely measure either one of those concepts and the reaming one can be readily known. For example, if the similarity between two documents is in the range 0.0 (least similar) to 1.0 (most similar), the distance between these documents can be calculated as:

$$Distance = 1.0 - Similarity$$

Conversely, similarity can also be calculated from a normalized distance (ranged 0.0 to 1.0) as:

$$Similarity = 1.0 - Distance$$

A number of methods can be used to determine the similarity (and hence distance) between two given documents. Before giving the definitions of similarities, let's first look at the concept of "weight" of each word.

In general, the weight w of a word in a document is a combination of its local weight l, which is specific to that document, and global weight g, which is common for all the documents in the corpus (Marcus, 2011).

$$w = l \times g$$

One of the most common forms for local weight is the term frequency tf, which means the frequency (count) of this word in that given document.

$$l = tf$$

One of the most common forms for global weight is the inverse document frequency idf, which is the logarithm (base 2) of the ratio of the total number of documents in the corpus N to the number of documents f in which the word occurs.

$$g = idf = \log_2 \frac{N}{f}$$

It should be noted that the inverse document frequency idf for each word is independent of the query and can be calculated and stored in advance.

Therefore, the weight of a word is calculated as:

$$w = tf \times idf = tf \times \log_2 \frac{N}{f}$$

Now, we can calculate the similarity between the query Q represented as a vector of the normalized weights of the words it contains: $(Q = [w_1^q, w_2^q, \dots, w_t^q])$ and a document D in the corpus also represented as a vector of the normalized weights of the words it contains: $(D = [w_1^d, w_2^d, \dots, w_t^d])$, where t is the number of all unique words in the corpus.

For the sake of clarity, let us denote $w_i^q = x_i$ and $w_i^d = y_i$, where $1 \leq i \leq t$.

The magnitudes of the vectors Q and D can be calculated as:

$$|Q| = \sqrt{\sum_{i=1}^{t} (x_i)^2} \quad \text{and} \quad |D| = \sqrt{\sum_{i=1}^{t} (y_i)^2}$$

It should be noted that $|D|$ can be pre-computed for all the documents in the corpus. The dot product of the two vectors Q and D is calculated as:

$$Q.D = \sum_{i=1}^{t} x_i . y_i$$

Some of the most common measures to calculate the similarity or the distance of Q and D are as follows:

Cosine Similarity:

$$Similarity(Q, D) = \frac{Q.D}{|Q|.|D|}$$

Jaccard Similarity:

$$Similarity(Q, D) = \frac{Q.D}{|Q|^2 + |D|^2 - Q.D}$$

Dice Similarity:

$$Similarity(Q, D) = \frac{2.Q.D}{|Q|^2 + |D|^2}$$

Normalized Euclidean Distance:

$$Distance(Q, D) = \sqrt{\frac{\sum_{i=1}^{t}(x_i' - y_i')^2}{2}}$$

where x_i' and y_i' are the normalized weights in the unit query vector $Q' = Q/|Q|$ and the unit document vector $D' = D/|D|$ respectively.

It should be noted that the indexing mechanisms discussed above are utilized in calculating the similarity of the query to every documents in the corpus.

For example, in inverted file indexing, for each word in the query (after removing stopped words and stemming), the posting list corresponding to that word is visited in order to calculate the values of $x_i.y_i$ for that word with respect to all the documents. In order words, the dot product $Q.D$ for every document is incrementally updated whenever a posting list corresponding to each word in the query is visited. After visiting all the concerning posting lists, the final value of $Q.D$ for every document is obtained. The term $|D|$ for every document has been usually pre-computed, and the term $|Q|$ for the query can be computed straightforwardly. Thus, at the final, based on the similarity measure of one's choice, the similarity of every document D in the corpus with respect to the query Q is obtained. Then, the documents are scored (ranked) according to their similarity values. The advantage of inverted file indexing is that we can compute the similarity between the query and the documents without actually visiting the documents

in the corpus but by merely visiting the index. This is particularly useful when the documents are of large sizes.

In the vector space model, we can elect to use a distance measure such as normalized Euclidean distance and use a distance-based indexing technique like iDistance (Jagadish et al., 2005) in order to select any given k number of nearest (i.e., most similar) documents using a k nearest neighbors (kNN) algorithm.

2.1.4 Evaluation of Retrieval Effectiveness

The effectiveness of a retrieval exercise with regard to a query can be evaluated with the metrics of precision, recall, and F-measure. Suppose that for a given query, we rank the documents in the corpus using a similarity/distance measure of our choice. Then we retrieve the top k highest ranking documents as our answer. Among those k documents, some of them (say r of them) are "actually" relevant to query, but some may be not, unfortunately. (Those documents which are actually relevant must be identified by human experts beforehand.) Suppose that in the corpus, there are a total of R documents which are relevant to the query. The measures of precision and recall are defined as:

$$precision = \frac{r}{k}$$

$$recall = \frac{r}{R}$$

Both precision and recall range between 0.0 and 1.0. The higher the precision/recall is, the better the retrieval system is. Precision and recall tell us about the two different perspectives of accuracy and coverage regarding the retrieval exercise. In order to reflect those two perspectives simultaneously, F-measure is introduced.

$$F = 2 \times \frac{precision \times recall}{precision + recall}$$

2.1.5 Other Information Retrieval Techniques

In addition to the core task of document retrieval, other tasks and techniques that fall under the broader area of information retrieval are as follows:

- **String matching:** to perform exact or approximate matching of one text (sub)string to another. This is useful in exactly pinpointing input search phrase(s) inside the documents that are highly ranked after executing a query (like Google does). String matching methods like Rabin-Karp algorithm, finite automata method, Knuth-Morris-Pratt algorithm etc. can be used for this task (Cormen et al., 2009).
- **Document categorization:** to automatically categorize a set of documents into two or more distinct categories (like news reports, articles, scientific papers, etc.). It is a supervision learning task in which the desired categories must be predefined and some sample documents for each category must be available in the first place (Sebastiani, 2002).
- **Document clustering:** to split a given set of documents into their respective groups. It is an unsupervised learning task in which the groups (categories) themselves are not predefined (Andrews and Fox, 2007). However, whether it is needed to predefine the required number of groups or not depends on the type of clustering algorithm used.
- **Duplicate document detection:** to detect the documents that are exactly the same or almost the same (Chowdhury et al., 2002). This is useful for storage space reduction, plagiarism detection, document versioning and control, etc.
- **Text data mining:** to extract a specific type of information from a single or a set of documents (Cohen and Hunter, 2008). For example, from a set of scientific papers on chemistry, all the chemical reactions mentioned in them can be extracted using text mining techniques.

2.2 Information Retrieval for Software Engineering

In this section, we will provide the concrete examples of how IR techniques can be used to improve the productivity of software developers in general and how they can facilitate a software engineering KM system in particular.

We first have to build a corpus for a piece of software by parsing its program source codes and extracting key words from other documents (Marcus, 2011). As generally described on Section 2.1.1, this involves removing the stop words in English, standard function library names, programming language keywords, etc. After that, another preprocessing job is carried out to standardize the variable names, etc. For example, the variable names of `max_count`, `MaxCount`, `MaximumCount` in multiple source code files are now unified to a standard one (say `MaxCount`). (If the software house has a strict standardized naming convention, this preprocessing job will be much easier.) Then, stemming is done to convert derived words into their respective base words (like converting `MaxCounter` into `MaxCount`). Likewise, Lexical analysis can be performed to break up text strings into words or tokens (like splitting `MaxCount` into two words: `Max` and `Count`). In addition to the source code files, other types of documents such as feasibility reports, requirement specifications, system designs, program designs, test scripts, test reports, user complaints, maintenance reports, etc. are also to be processed and stored into their respective corpuses or a centralized corpus. After building the corpus(es), any type of indexing of one's choice, as described in Section 2.1.2, is applied on it with a view to speeding up the processing of queries in the future.

2.2.1 IR for Software Engineering in General

In general, IR can be applied to some of the important problems in SE like the *concept localization* problem for discovering a point of change, the *traceability link recovery* problem for finding the links of documents associated with a proposed change in requirement, the *bug triage* problem for verification of bugs, assignment of severity level, and recommendation of relevant developer, etc., and the *software clustering* problem for organizing similar software entities into intrinsic groups (Marcus, 2011).

Concept Localization: It means "discovering human oriented concepts and assigning them to their implementation instances within a program" (Biggerstaff et al., 1993). Concept location is needed whenever a modification is to be made to software at various design levels and/or at source code level. Thus, it must be within a well-defined context and level of abstraction. The locations of a particular concept (e.g., `Change Input Method`) may be different for the task undertaken by a high-level system designer and that by a developer. The user usually formulates a query and executes it to get the ranks of the corpus's documents that are relevant to his/her task in hand. Then, he/she can examine the results and make changes to the affected original documents accordingly. This procedure is repeated a couple of times with different queries. Concept location is also useful in determining the dependency and coupling of codes.

Traceability Link Recovery: It means "the ability to describe and follow the life of a requirement, in both forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases.)" (Gotel and Finklestein, 1994). This problem is related to the above concept location problem but the objective is different. Since it is desirable to discover as many documents that involve in the link in question as possible, a higher recall rate is much desirable. The user can review the traceability link reported by the IR system, validate it, and report his/her feedback on its relevance. Traceability link recovery is also useful in impact analysis of software bug/failure and in identification of reusable software components.

Bug Triage: When a software bug occurs, the project in-charge has to verify the incoming bug report, assign a severity level to it, and allocate a developer to solve it (Marcus, 2011). It can be started by analyzing the contents of the bug report. The sub-problems for the bug triage process are: duplicate bug detection, developer recommendation, assignment of severity level, and detection of security bugs. We can use string matching, document classification, clustering, and duplication detection tools in order to automate this process.

Software Clustering: The purpose is to group software entities into clusters such that the entities in the same cluster are similar to each other, and those in different clusters are different. Document clustering techniques (Andrews and Fox, 2007) can be utilized for this purpose. The clustering results can be evaluated using cluster quality measures like silhouette analysis (Rousseeuw, 1987). Software clustering is useful for software architecture recovery, identifying the topics implemented, detecting software clones, software re-modularization, program comprehension, etc. (Marcus, 2011).

2.2.2 IR for Proposed Constructive Knowledge Management Model

The primary role for an IR system in our proposed constructive knowledge management model for software development is to efficiently retrieve the knowledge mainly from the knowledge repository and its ontological metadata in some cases. Most pieces of knowledge stored in the repository will be in the form of highly standardized, concise and restricted-vocabulary textual information. Therefore, standard IR corpus building, indexing strategies and query processing techniques can be used to facilitate efficient and effective retrieval by an organizational knowledge user. For the retrieval from the ontological metadata, ontology-based retrieval methods like Vallet et al. (2005) could be used.

During the testing period, the effectiveness of the retrieval results for each query can be manually verified with regard to their actual relevance to the topic in question and then evaluated using the metrics of precision, recall, and F-measure. We can experiment with different combinations of IR models (like vector space or latent semantic) and similarity/distance measures (like Cosine, Jaccard, Dice, or normalized Euclidean), and select the one which gives us the best results.

3. Conclusion

The paradigm shift of Knowledge Management is growing very firstly in many of organizations nowadays. The fundamental concepts and processes discussed here reflect of growing shift in management thinking and practice today. This paradigm shift is from traditional management concept which is based on command and control to new

concept of management which is more concerned with developing, supporting, connecting, leveraging and empowering employees as knowledge workers.

In Software Engineering environment, Knowledge Management is of interest to many software professionals and researchers and is being tested by many organizations as a potential solution repository related to software development problems. As we can see recent developments in IT definitely enable sharing documented knowledge independent of time and space. We can foresee that there will also be support for capturing and disseminating knowledge in various formats, enabling organizations and individuals to share knowledge on a world-wide scale in the future.

In addition to knowledge management, information retrieval also plays a crucial role in Software Engineering. IR can be used both as a general-purpose tool to improve the productivity of developers or as an enabler tool to facilitate a Knowledge Management system.

References

1. J. Andrade, J. Ares, R. García, S. Rodríguez, and S. Suárez, “A reference model for knowledge management in software engineering,” *Engineering Letters*, 13, 159-166, 2006.
2. N. O. Andrews and E. A. Fox, *Recent Developments in Document Clustering*, Technical Report TR-07-35, Virginia Polytechnic Institute and State University, 2007.
3. R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval: The Concepts and Technology behind Search (2nd Edition)*, ACM Press, 2011.
4. E. Bertino, B. C. Ooi, R. Sacks-Davis, K. L. Tan, J. Zobel, B. Shilovsky and B. Catania, *Indexing Techniques for Advanced Database Systems*, Kluwer Academic Publishers, 1997.
5. T. J. Biggerstaff, B. G. Mitbander, and D. Webster, “The concept assignment problem in program understanding,” in *Proceedings of the 15th international conference on Software Engineering (ICSE’93)*, pp. 482-498, 1993.

6. A. Birk, D. Surmann, and K.-D. Althoff, “Applications of knowledge acquisition in experimental software engineering,” in *Proceedings of 11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW’99)*, pp. 67-84, 1999.
7. F. O. Bjørnson and T. Dingsøyr, “Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used,” *Information and Software Technology*, 50, 1055-1068, 2008.
8. F. Y. Y. Choi, “Advances in domain independent linear text segmentation,” *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL’00)*, pp. 26–33, 2000.
9. A. Chowdhury, O. Frieder, D. Grossman, and M. C. McCabe, “Collection statistics for fast duplicate document detection,” *ACM Transactions on Information Systems*, 20, 171-191, 2002.
10. K. B. Cohen and L. Hunter, “Getting started in text mining,” *PLoS Computational Biology*, 4, e20, 2008.
11. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (3rd Edition)*, MIT Press, 2009.
12. T. H. Davenport and L. Prusak, *Working Knowledge: How Organizations Manage What They Know*, Harvard Business School Press, 1998.
13. S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, “Indexing by latent semantic analysis,” *Journal of the Society for Information Science*, 41, 391-407, 1990.
14. G. Fischer and J. Ostwald, “Knowledge management: Problems, promises, realities and challenges,” *IEEE Intelligent Systems*, 16, 60-72, 2001.
15. O. C. Z. Gotel and C. W. Finklestein, “An analysis of the requirements traceability problem,” in *Proceedings of the 1st International Conference on Requirements Engineering (ICRE’94)*, pp. 94-101, 1994.
16. H. V. Jagadish, B. C. Ooi, K. L. Tan, C. Yu, and R. Zhang, “iDistance: An adaptive B+-tree based indexing method for nearest neighbor search,” *ACM Transactions on Data Base Systems*, 30, 364-397, 2005.

17. R. LaBrie and R. S. Louis, “Information retrieval from knowledge management systems: Using knowledge hierarchies to overcome keyword limitations,” in *9th Americas Conference on Information Systems (AMCIS’03)*, pp. 2552-2563, 2003.
18. I. Nonaka and H. Takeuchi, *The Knowledge Creating Company*, Oxford University Press, 1995.
19. C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.
20. A. Marcus, “Information Retrieval Methods for Software Engineering,” in Canadian Summer School on Practical Analyses of Software Engineering Data (PASED’11), Montreal, Quebec, Canada, 2011. URL: http://pased.soccerlab.polymtl.ca/materials/june-18-2011/PASED-06.18.2011-Slides.pdf
21. P. J. Rousseeuw (1987). “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Computational and Applied Mathematics*, 20, 53-65, 1987.
22. H. Samet, *Foundations of Multidimensional and Metric Data Structures*, Morgan Kaufmann Publishers, 2006.
23. R. Sanchez, “Creating modular platforms for strategic flexibility,” *Design Management Review,* 15, 58-67, 2004.
24. R. Sanchez, “Managing knowledge into competences: The five learning cycles of the competent organization,” in *Knowledge Management and Organizational Competence,* Oxford University Press, pp. 3-37, 2001.
25. R. Sanchez, “Managing articulated knowledge in competence-based competition,” in *Strategic Learning and Knowledge Management,* John Wiley and Sons, pp. 163-187, 1997.
26. R. Sanchez, A. Heene, and H. Thomas (Editors), *Dynamics of Competence-Based Competition: Theory and Practice in the New Strategic Management,* Elsevier Pergamon, 1996.
27. F. Sebastiani, “Machine learning in automated text categorization,” *ACM Computing Surveys*, 34, 1-47, 2002.

28. R. Sindhgatta, “Using an information retrieval system to retrieve source code samples,” in *Proceedings of the 28th International Conference on Software Engineering (ICSE’06)*, pp. 905-908, 2006.
29. D. Vallet, M. Fernández, and P. Castells, “An ontology-based information retrieval model,” in *Proceedings of the 2nd Annual European Semantic Web Conference (ESWC’05)*, pp. 455-470, 2005.
30. J. P. Walsh and G. R. Ungson, “Organizational memory,” *Academy of Management Review,* 16, 57-91, 1991.