Software Development and Reuse as a Knowledge Management Practice

Abstract: Software development is usually referred to as a knowledge intensive practice. In fact, companies involved in software development are said to be the most clear example of the companies whose revenue and value is based in the creation, representation and modeling of knowledge.

The software development process starts with the capture of some requirements provided by the final users or by the project stakeholders. Taking these requirements as a basis, the development teams must cross the chasm from the text-based requirements to their representation in a specific programming language.

To move from the textual representation of the software requirements to the final code, different steps and artifacts must be created. These artifacts also represent the knowledge embedded in the functional specifications, and are an intermediate step to reach the final representation: the programming code.

The author analyses the impact of knowledge management in software development processes, and describes the possibility of using a shared pool of code to make possible the sharing of the knowledge gained.

A prototype tool to create and maintain the repository of code and link this code to functional specifications and any other artifacts (documents, models, reports, etc.) is described. This tool will link together the knowledge created at the different levels (business analysis, analysis, design, coding, etc.) in the development process to make the reuse of code easier.

1. Knowledge Management

Today, knowledge management (KM) has become a key issue for any organization regardless its size and business activity sector. Companies recognize KM as an important practice to achieve competitive advantages, as KM programs can give them the possibilities of:

a) increasing innovation; b) cutting the time needed to design and market new products and services; c) ensuring the effectiveness of your employees; d) reducing the costs of the training and external consultancy and e) incrementing the motivation and participation of the employees.

This has led to an increasing number of projects focused on the development of KM strategies. But the first challenge companies must face when establishing a KM program is to clearly define the scope of the project and give answers to a sometimes difficult question: what do we mean when we are talking about knowledge management?

KM means putting in action different practices with a single purpose: make the creation and transfer of ideas and knowledge easier. More concretely, KM means:

a. Gathering best practices and the most efficient work procedures, and deploy them throughout the whole organization.
b. Sharing the ideas and the tacit knowledge available in the brains of the employees, clients and partners.

c. Making all this knowledge and expertise explicit.

d. Creating knowledge bases or repositories where the explicit knowledge is linked to work processes and daily tasks.

e. Ensuring that all the employees can access the knowledge they need to complete their tasks, work and assignments.

f. And finally, encouraging employees to use the knowledge, and measuring this reuse.

2. Knowledge management and software development

Software development is usually referred as a knowledge intensive industry. In fact, when developing software, analysts and programmers must identify the knowledge about business practices provided by the stakeholders and future users of the systems, make this knowledge explicit by means of functional specifications or any other kind of documents, and transfer all this information to other people in the production chain that will take them as a basis to create additional artifacts until the final coding and deployment of the software.

During this process, knowledge must be made explicit and shared between people. Different artifacts are used to represent it: textual information, graphics and diagrams, formal languages, programming languages, etc. These artifacts can be seen as different levels of knowledge representation.

Most companies have adopted software development methodologies to provide workers with guidance and rules they must follow to make the transition from one level to another.

These methodologies try to standardize the best possible way of creating software and moving from the initial specifications to the final code.

However, the success of these methodologies is not clear. Some people are skeptical about them, and the software development process is still considered to be something difficult to manage: an anarchic process that depends, in a great extent, on the personal commitment and knowledge of the individual programmers.

People involved in software development consider these methodologies as an additional and unnecessary workload that do not supply any real improvement in the development of the applications.

These considerations are due to a lack of knowledge of the advantages that we can achieve in the near future once we have spent a little time representing knowledge. In fact, the importance of the methodologies goes beyond the document types they recommend to write down. What a methodology really provides is a systematic way to represent and organize both functional and technical knowledge.

In general, although methodologies are sometimes used to represent knowledge, little effort is currently being done to improve the sharing of this knowledge. So, people think of methodologies as overwhelming papers instead of a knowledge sharing tool.

As a result, in order to develop better software and help manage current and future development projects, some techniques and practices developed in the KM area should be borrowed and merged with the knowledge representation techniques proposed by the traditional software development methodologies.
3. The role of documentation in KM

The success of any approach aimed to improve the management of knowledge in a software development company will depend on our ability to:

a) Capture and make explicit the working knowledge created by the community of developers and organize it in an efficient way

b) Provide context to this knowledge by means of different artifacts that cross the gap between the requested functionality, the code, and the intermediate documents created with modeling languages such as the UML (Unified Modeling Language).

c) Reward the reuse of knowledge

Software documentation plays a key role in the proposed KM strategy. Although sometimes deprecated and considered an inconvenience, one of the main steps in any KM strategy consist of writing down the knowledge available in our brains.

This process, referred to as externalization in Ikujiro Nonaka's classic book *The Knowledge-Creating Company* points out the need to write down what we know and move it to an organized and shared repository of documents.

This makes software documentation important, as it must be understood under a new perspective: documents should be seen as the tool we use to transfer knowledge and make this knowledge available for further reference.

4. Problems due to the lack of documentation

But regardless their impact on a KM strategy, the lack of documentation is the origin of several problems in a software development organization: *Impression of low quality.*

Systems and system development processes cannot be audited if no design documents are available. External evaluators and inspectors consider the lack of documentation as a symptom of insufficient organization and inefficient work processes. If the system documentation is not up-to-date, the results of the audits and the subsequent evaluation will be negative to the system and the whole organization.

*Higher development and maintenance cost.*

Finding defects in the early phases of software development (that is to say, in the specifications of the requirements and the design instead of in the programming code), significantly decreases the costs of defect removal.


Without documentation, the only reliable and objective information is the source code itself. In such a situation, programmers must spend a great amount of time trying to understand the system's functionality by exploring its source code.

Experience shows that the time spent by maintainers studying source code is 3.5 times greater than the time they should spend studying documentation *(Source SEI CMU)*

So, documentation is critical for software engineers and technical managers responsible for the evolution of the software.

4.2. System integration.

Software applications are not isolated pieces; they need to talk to each other to better support critical processes. The evolution of IT makes unpredictable the
future requirements of any company, as well as the needs to interface with coming technologies and applications.

Documentation ensures the possibility of extending the system with interfaces to other applications provided by third parties. The lack of documentation will force the development teams to face costly re-engineering projects when an integration with other system is requested.

From a business perspective, the difficulty found when designing interfaces to other systems by a lack of accurate documentation also have a negative impact in the time-to-market and reliability of future IT projects. In fact, even the ultimate application just developed will become legacy in the near future.

4.3. Corporate amnesia.

Employees' turnover is one of the most important problems for software development projects, and although it is true that nobody is irreplaceable, there are costs associated with employee turnover. Errors made by novice programmers who do not have a detailed description of the software application reduce the efficiency of the whole system, and the quality of the code degrades as more and more people updates it without the guidance of documents.

Problems in team communication. In any software development project, communication is a key factor to obtain success. Documentation helps companies improve communication between the different teams and developers involved on the project. This factor is even more important in the case of companies and projects where external consultants or programmers are hired in a temporary basis.

5. Problems with the traditional approaches to software documentation and proposed solutions

There are several reasons that detractors of software documentation put forward:

a) Documentation does not make the code more maintainable.
b) Documents become obsolete as soon as they are created.

The reason of these problems are that usually documentation does not contain clear links between the functionality requested by the Client and the system code and technical descriptions. As a result, people need to invest a lot of time reading a huge amount of "paper", and even after reading this overwhelming documents, they cannot gain a clear understanding on how a specific functionality has been implemented, as documents are not related to each other or to the real implementation.

To solve these difficulties, we need to offer a documentation set that links functional specifications and list of features to the documents explaining their technical implementation and to the programming code. For example, if we apply the UML modeling language – the current de facto standard in software modeling – documents must be clearly organized and linked by means of roadmaps that lead from the use cases that describe the requested functionality to the technical diagrams and documents showing its implementation (class diagrams, reference to methods, components diagrams, etc.)

Linking the functional specifications with the technical design and with the repository of reusable and shared code will give rapid answers to the questions developers frequently ask themselves: what classes are involved in this
functionality? which method contains the code to validate the correctness of this process?

With this documentation, developers can be confident that they are making changes in the right place.

6. KM Project Implementation

The first step in the KM strategy is to recognize the importance of documentation, and the need to link documents created through the different steps in the software development process. These steps will change depending on the development methodology each organization applies.

The main target of the project we describe in the following sections is to reduce the time needed to develop a specific functionality by reusing programming code. This is an important source of benefits in those companies developing applications of the same type.

The second step in the proposed strategy is the creation of a repository of documents and source code. In this repository, source code will be linked to the functionality it implements, that is to say, to the documents and diagrams that describe this functionality.

To create the repository, we will complete these tasks:

Documentation is first created by means of UML diagrams and attached documents. UML modeling tools such as Rational Rose or TogetherSoft offer the possibility of generating documentation in html format. This documentation will contain all the diagrams and explanations included in the UML model. This documentation will be the basis of the subsequent steps.

Then, code is documented by developers, who will add a detailed description for each class, method, function, etc. they code. This comments are written inside the code. To distinguish the comments explaining the purpose and structure of the code from any other comments, a special markup is used. Developers should also add some metadata that will be used later to link the code to functional specifications or set of features.

By means of an automated process, source files containing code are parsed in a regular basis to extract the comments containing the metadata and documentation. For each class, operation and method processed, an html page is generated. This page will contain comments, metadata, and the source code itself. Metadata entered by developers are converted to html meta properties, to make possible their use in queries.

This process is similar to the JavaDoc tool, released by Sun Microsystems, to automatically extract comments and documentation from Java code and generate documentation in html format. A good explanation on how to accomplish this process and automate the extraction of comments in VB projects can be found in the current literature (Schultes, 2001).

The html pages are automatically moved to a repository where they are indexed. Indexing the full-text as well as the metadata properties gives the possibility of accessing the code used to implement specific processes or use cases, activities in a use case, product features and any business rule specified in the functional specifications.
Finally, the last step consist of linking the html pages generated from the code to the functional documents and diagrams generated in the first step. Pages are linked together by means of a navigable table of contents.

This table of contents contains references to html pages describing use cases, activities, sequence diagrams, classes and components. To update this table of contents with all the necessary links, an automated process was developed.

This process reads the metadata available in the html page that describes each piece of code and identifies the name of the class or method. As this information is also available in the UML models, it is not difficult to insert the links in the appropriate location.

As a result, developers can access an updated repository of source code linked to the functional documentation. This enables them to search for the implementation of specific functions or features. Full-text and metadata searching capabilities and the navigable table of contents help them locate the code and functions.

7. Additional features and next steps

The availability of a shared repository of code is the first step toward a knowledge management and reuse strategy. But sometimes, two programmers may code the same function in different ways. This may affect the quality of code, and errors can be propagated inadvertently.

To avoid this problem, a separate step that must be implemented is the revision of code by senior programmers. Senior programmers will be in charge of reviewing the code, fixing possible bugs, and identifying the best approach to solve a problem when two or more alternatives are available.

This is also a best-known practice in KM programs. The role of the knowledge guardian or subject expert who reviews and validates contributions to the knowledge base can also be applied in the software development industry. In fact, this is not something new, as the role of the senior programmer who reviews and validates the code written by other programmers has always been recognized as a must-have in different methodologies and software development practices.

The implementation of the validation process adds the requirement of dealing with two possible scenarios:

In the first one, code is published once it has been reviewed. So, we need to distinguish between validated and not-validated contributions. This ensures a higher quality of the code available in the repository, but makes knowledge transfer slower.

In the second scenario, code is moved into the repository automatically, with no prior validation. If any inconsistency is identified, the programmer in charge of maintaining the repository will have the possibility of rejecting a piece of code from the repository when a better alternative is already available. People who have reused this code should be automatically notified about this change.

Other functions that go beyond the current proposal are the possibilities of creating and managing discussion forums, adding comments or suggestions to code in the repository, or scoring the existing code. This will be an important issue to measure the effectiveness of the code, its reuse and the success of the whole implementation.
Other methods to measure the success of this KM program might consist of the counting of access to the repository or the LOC (lines of code) shared or taken from it.

Anyway, although any of these approaches might be used, we strongly recommend a different approach based on the scoring that developers assign to the implementation suggested by other colleagues, as the main purpose in the KM strategy is leading to valid solutions.

References