TECNOLÓGICO
DE MONTERREY®

Improving Software Engineering: A Knowledge Management Approach

Marcos Eliud Guevara Camacho, Carlos Adrián Solares Lozano

December, 2004

Contents

Abstract

Knowledge Management is a mature trend for the new wave of enterprise innovation. It permits to storage the intangible assets of a company to standardize processes and to potentialize improvements for better performance. Software Engineering is the science for develop today's complex computer programs. It provide diverse methodologies to fully accomplish projects in terms of requirements, time or budget. This paper presents the foundation of Knowledge Management and Software Engineering. Then a framework is proposed for applying KM techniques to development of software.

1 Introduction

Knowledge Management (KM) is a mature trend for the new wave of enterprise innovation. It permits to storage the intangible assets of a company to standardize processes and to potentialize improvements for better performance.

Improvements are achieved through the generation of value by using intangible assets as sources of knowledge. Therefore all of the value is added to a process relies heavily in the quality of the knowledge that is used. Nowadays it can be seen that Knowledge Management is becoming a fundamental part of the comparative advantage that big companies have.

Software Engineering (SE) is the science for develop today's complex computer programs. It provide diverse methodologies to fully accomplish projects in terms of requirements, time or budget. The abstraction component of Software Engineering makes it unique from traditional manufacturing engineering fields. Also, the problem domain of software engineering encompasses almost all domains in the real world and it is focused on transform the information processing and intelligent parts of the conventional physical products into software [32].

This paper presents the foundation of Knowledge Management and Software Engineering. Knowledge Management patterns will be identified through the Software Engineering part. Then a framework is proposed for applying Knowledge Management techniques to development of software.

2 Software Engineering

Software Engineering is a unique discipline with philosophical, mathematical and managerial foundations based on interdisciplinary knowledge. Gartner, an important consulting firm in Information Technology (IT) has predicted that 50% of business will cut out from consulting teams that will not apply Software Engineering best practices [17]. For the future, Software Engineering is a mandatory approach for organizations that will implement software to support their most important operations.

1 Introduction

Knowledge Management (KM) is a mature trend for the new wave of enterprise innovation. It permits to storage the intangible assets of a company to standardize processes and to potentialize improvements for better performance.

Improvements are achieved through the generation of value by using intangible assets as sources of knowledge. Therefore all of the value is added to a process relies heavily in the quality of the knowledge that is used. Nowadays it can be seen that Knowledge Management is becoming a fundamental part of the comparative advantage that big companies have.

Software Engineering (SE) is the science for develop today's complex computer programs. It provide diverse methodologies to fully accomplish projects in terms of requirements, time or budget. The abstraction component of Software Engineering makes it unique from traditional manufacturing engineering fields. Also, the problem domain of software engineering encompasses almost all domains in the real world and it is focused on transform the information processing and intelligent parts of the conventional physical products into software [32].

This paper presents the foundation of Knowledge Management and Software Engineering. Knowledge Management patterns will be identified through the Software Engineering part. Then a framework is proposed for applying Knowledge Management techniques to development of software.

2 Software Engineering

Software Engineering is a unique discipline with philosophical, mathematical and managerial foundations based on interdisciplinary knowledge. Gartner, an important consulting firm in Information Technology (IT) has predicted that 50% of business will cut out from consulting teams that will not apply Software Engineering best practices [17]. For the future, Software Engineering is a mandatory approach for organizations that will implement software to support their most important operations.

2.1 The history of the Software Engineering Process

The term emerged in 1969 during a conference of NATO [18] by Fritz Bauer as follows:

> *Software Engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.*

The NATO conferences were originated because of the main problems of military systems projects: lack of quality, over-budgeted and out of time.

The main objective of Software Engineering is now to avoid a Software Crisis. Software Crisis was a stage of time in which software development did not have a methodology nor tools to deliver products on-time and on-budget. As computer science was delivering new programming tools and algorithms, software engineering took advantage of other fields like management. The term of project was adjusted to the software development lifecycle.

In 60's decade several components were discussed as part of Software Engineering, but in the last three decades, they have been refined, updated and extended. In the rest of this section, each one will be presented and finally those state-of-the-art techniques will be clarified.

- A Development Lifecycle.
- Methodologies for construct software.
- Tools for Software Engineering.
- Processes that will combine a methodologies with tools.

2.2 The Development Lifecycle

A development lifecycle is s subset of the software lifecycle. The software lifecycle is the set of stages in which a software passes. Since its inception through its elimination,

software has a subset of developing stages. The development lifecycle takes charge of this subset in the following stages: *Analysis, Design, Coding, Testing & Deployment*

2.2.1 Analysis

Analysis is a critical stage in software development. It comprehends since the requirements establishment to a Software Specification. In specific, analysis stage has the major contact with persons that will benefit from software. It takes in count their considerations, agreements but basically requirements about the software product.

Requirements Engineering. An increasing problem caused by a poor analysis appears in following stages and it costs more in terms of time and money. An special trend that most of methodologies have accorded is Requirements Engineering. Stressing the antecedent that lack of analysis leads to software failure requirements process took an importance and engineering principles were applied. Requirements engineering now covers the analysis stage in four tasks: Requirements elicitation, analysis, verification and management.

Sommerville [29] explains a basic concept of elicitation. First the Analyst gets a full comprehension of the problem domain the system will overcome (a supermarket, a bank, a shuttle, etc.) sometimes, it will need help from domain experts involved in software development.

2.2.2 Design

Design phase involves taking the requirements and devising a plan and a representation to allow the requirements to be translated into source code . David Budgen recognizes three parts: representation, process and heuristics. The representation part consists in notations (some are parts of tools) that will describe a design model. The process part indicates how the model will be developed or expressed. Process is greatly influenced by And finally, design heuristics serve as "rule of thumb" for guidance in particular problems [4].

Software Architectures Architectures emerge as a model of interactions of all components in a given software. Commonly, architectures are presented in layers (e.g. as the OSI model). And they encompass the relations with the problem domain, software components, hardware infrastructure, information transformation and strategic deployment. These architectures help designers, Client and Stakeholder understand the software product before it is manufactured [18].

Software Development Methodologies influence in a deep manner the process and representation part. These can short the time dedicated to design or can reset the design phase. Human creativeness is also a factor of influence to the overcome of design. Consider for example a problem that has established a set of goals (requirements). When two persons tackle the problem, they will get to the goals, but in different manners (design).

2.2.3 Coding

Coding sometimes is confused with construction. Construction embraces three stages: design, coding and testing. Coding refers to the time in the development cycle in which design specifications are transformed in software. When coding is started, requirements are evaluated and the elements of the system are envisioned as software units. Then design and architectures glues these software units into a unique software. Finally, programming strategies are established. A programming plan must be present to manage resources and overall project status [5].

Objet-Oriented Programming The object-oriented model for software development has become more attractive for software development in organizations and also for research purposes. Pokkunuri [23] set in a research the time line of programming languages. An interesting fact is that object-oriented programming languages were the base from computer science to create methodologies and requirements processes.

2.2.4 Testing

Called also validation, testing is the phase in which today's projects invest 40% of resources. The time of testing is influenced by how well the preceding stages were

performed. During the testing stage, the fundamental element is a test case. Test cases, are the checklists of characteristics of the software system. These cases are often realized in parallel with the design stage. They have an input and an expected output of what the system should do. Software Development Methodologies have a medium influence in terms of speeding the testing phase.

Quality Assurance. Testing is a minor part of overall Quality Assurance (QA). Quality Assurance has been well established by Pressman as follows: *"Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software."* [25].

2.2.5 Deployment

Software Deployment has been a phase most uncontrolled in practice. It manages activities from finalizing testing to formal usage from clients. Even tough software engineers have formal strategies to deploy software research is not as public as research in other phases.

Levels of Deployment Caupaye and Estubelier identified three levels of deployment: enterprise, process and user [7]. Enterprise level includes all the environment policies. Process level extends to specific tasks to install the software (e.g. A NASA Software for shuttles) and User level takes in charge of the learning-curve of the user with the software. Software development methodologies have a minor effect on this stage, in the way that approaches to this stage may be shared between methodologies.

2.3 Software development methodologies

Software development methodologies are mature procedures of (1) how to manage each stage and (2) Strategic sequence of of the software development lifecycle.

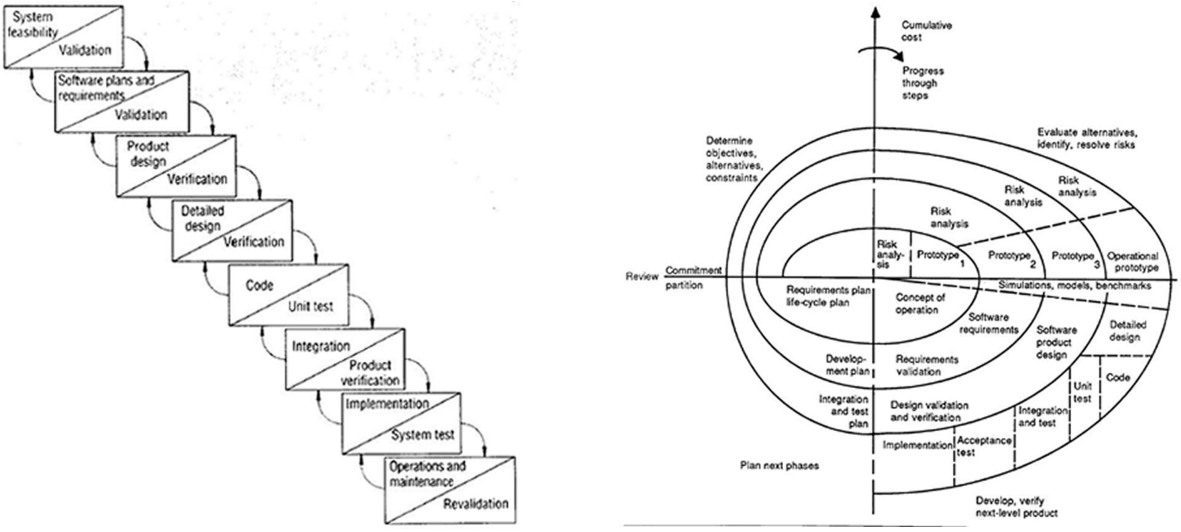

Figure 1: The Waterfall and Spiral methods [3].

2.3.1 Waterfall method

In previous work [10], the authors of this article have presented the waterfall method: "This method, also called the "conventional" software process, divides the whole process into subprocess and before to continue to the next phase it is needed to fully complete the actual phase. In practice, this process overlaps and shares most of the information. These methods are still used in some projects when the risks are most controlled." (see figure 1)

2.3.2 Spiral method

For the spiral method: "It was proposed by Boehm in 1988 [3] and its widely known. This approach represents the succession of activities with a retrospective from one activity to another (see figure 1). With this, each cycle could have a deliverable prototype [29]. The quadrants refers to:

1. *Determine objectives.* In this phase, the constraints of the product are identified. Also a detailed plan is established.

2. *Risk evaluation and reduction.* Several solutions are developed to counteract the

risks of software development.

3. *Development and validation.* Another method is used to develop and validate the software. This phase occurs after the risk evaluation and reduction phase.

4. *Planning.* At this stage the team takes a decision wether to continue to the next cycle or to finish the project.

2.3.3 Rapid prototype method

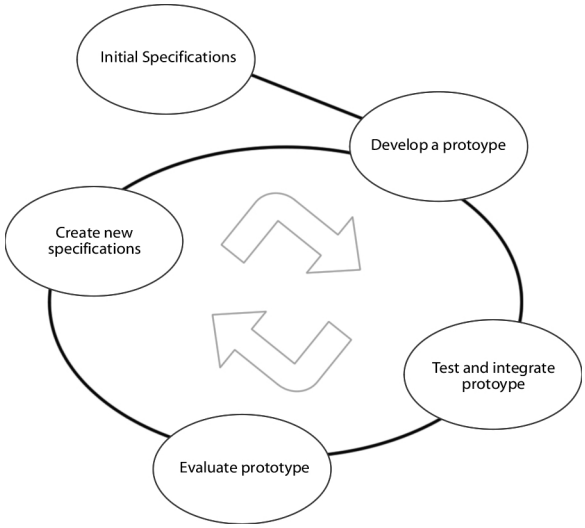

Figure 2: The Rapid Prototype method.

Rapid prototype methods are based on an iterative process and require the creation of one or more partial/incremental solutions. The primary assumption is that the complete requirements specification of the software is not yet finished. Instead, a software is developed incrementally, within the specification developing along with the software itself.

2.4 Tools for Software Engineering

Many tools have been proposed for each stage of the development lifecycle. Others depend on specific software development methodologies. Even tough there is a range

of tools and software in the industry, two of them are important for this work: *The Unified Modeling Language and CASE tools.*

2.4.1 Unified Modeling Language

The Unified Modeling Language is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. It was created by Booch, Rumbaugh and Jacobson. Each one combined their own methodologies and also integrate past diagram styles like the Entity-Relationship style. Since 1997 is part of the Object Management Group as a standard language for blueprints of software in the design stage.

UML has a broad approach in different phases of the development lifecycle. It is more related to the object oriented programming language. The example above is a class diagram for a Java System. UML has a foundation on views from Stakeholder, developers and Database designers.

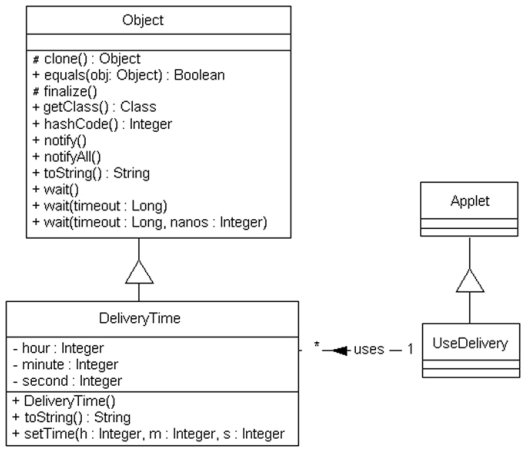

Figure 3: UML Class Diagram example [30].

2.4.2 Computer-aided Software Engineering Tools

Computer-aided Software Engineering Tools aid the development lifecycle. Since 1980's, the software industry tried to automatize all tasks; however, the collaborative part of

development has diminished its potential. In a development group of 50 persons, is widely used.

2.5 State-of-the Art of Software Engineering

Most of the advances in Software Engineering attack open issues in development methods and tools. A hype cycle from Gartner Research group shows the new advances in Software Engineering (or Application Development). In figure 4 the status of advances is presented as of July 2004. For this work, *Aspect-Oriented Software Development, Service Oriented Architecture, Agile Methodologies* will be analyzed. *Collaborative tools for the software development lifecycle and Metadata management will be presented as part of the framework.*

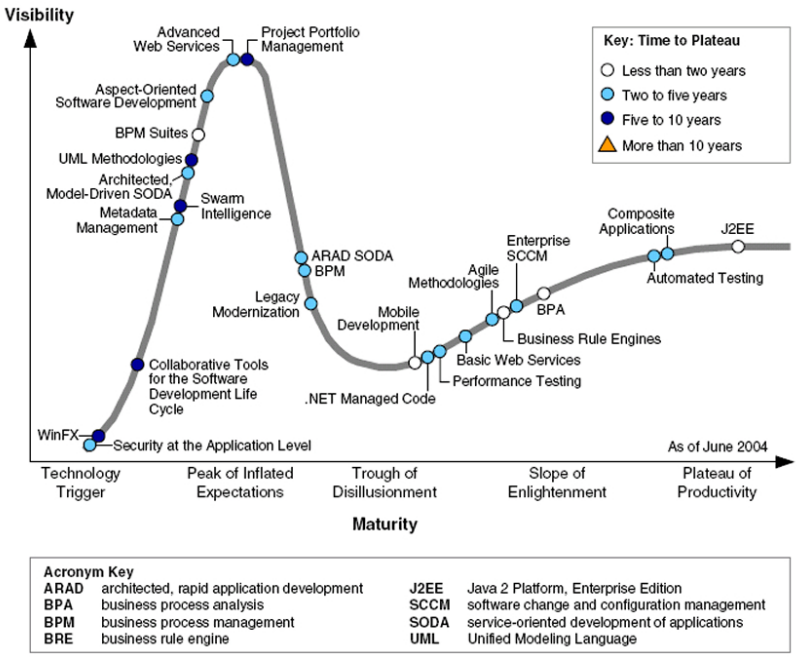

Figure 4: Hypecycle for Application Development [8].

Service-Oriented Architecture Service-Oriented Architecture (SOA) is focused on the abstraction of business services, in which a computing process is made by a

computing entity called service. In the Internet era, Web Services are integrated to create new applications and for enterprises, to integrate components of other partners. An introduction was given in 2003 by Papazoglou [21] with the diagram in figure 5.

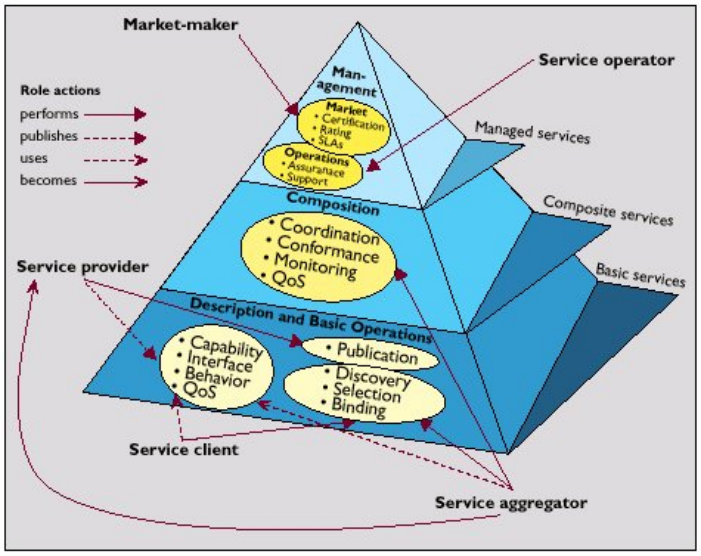

Figure 5: Extended service-oriented architecture: Service layers, functionality, and roles [21].

Aspect-Oriented Software Development Aspect-Oriented Software Development is a new trend for the Coding stage of the development lifecycle. It extends the Object-Oriented programming while specialize concerns about the software. It refers to the ability to identify, encapsulate, and manipulate those parts of software that are relevant to a particular concern (concept, goal, purpose, etc.). Concerns are the primary motivation for organizing and decomposing software into manageable and comprehensible parts. Many kinds of concerns may be relevant to different developers in different roles, or at different stages of the software lifecycle. As Elrad explains [9]:

> *It is based on the idea that computer systems are better programmed by separately specifying the various concerns (properties or areas of interest) of a system and some description of their relationships, and then relying*

on mechanisms in the underlying Aspect-Oriented Software Development environment to weave or compose them together into a coherent program.

Agile Methodologies Agile methodologies respond to a need in Software Engineering to deliver projects on time. Criticism has been detected to formal theory of methodologies and in recent years, development groups experiment with new methodologies presented by Radding [27]. An further reading of this paper will be a Knowledge Management Specialization in Agile Methodologies presented by Holz [12].

- **Extreme Programming.** A definition of its author is: "Extreme Programming is a lightweight design method developed by Kent Beck, Ward Cunningham, and others. After notable successes, XP has been generating huge interest, and no small amount of controversy. Much of the interest stems from XP's pragmatic approach to development. Key practices include pair programming, writing tests upfront, frequent refactoring and rebuild, continuous integration and testing. Key principles incremental and iterative development, working with the simplest solution, cutting out extraneous documentation, and collective code ownership" [2].

- **Scrum.** A methodology based on component-based development. It takes the advantages of rapid prototype approach and its popular among the industry. More recently it has been searched by the research community. Principal sources of further reading are in [28, 11]

- **Feature Drive Development.** Also called Requirement-Driven Development. Is foundation resides in model early requirements and models actors, goals and dependencies. Architectural design is always based on subsystems. A design language has been developed for this methodology called Tropos. Further reading is available by Castro and others [15, 14].

3 Knowledge Management

Nowadays, we read the word Knowledge Management in nearly every important business magazine. That is because, Knowledge Management has changed the way that

on mechanisms in the underlying Aspect-Oriented Software Development environment to weave or compose them together into a coherent program.

Agile Methodologies Agile methodologies respond to a need in Software Engineering to deliver projects on time. Criticism has been detected to formal theory of methodologies and in recent years, development groups experiment with new methodologies presented by Radding [27]. An further reading of this paper will be a Knowledge Management Specialization in Agile Methodologies presented by Holz [12].

- **Extreme Programming.** A definition of its author is: "Extreme Programming is a lightweight design method developed by Kent Beck, Ward Cunningham, and others. After notable successes, XP has been generating huge interest, and no small amount of controversy. Much of the interest stems from XP's pragmatic approach to development. Key practices include pair programming, writing tests upfront, frequent refactoring and rebuild, continuous integration and testing. Key principles incremental and iterative development, working with the simplest solution, cutting out extraneous documentation, and collective code ownership" [2].

- **Scrum.** A methodology based on component-based development. It takes the advantages of rapid prototype approach and its popular among the industry. More recently it has been searched by the research community. Principal sources of further reading are in [28, 11]

- **Feature Drive Development.** Also called Requirement-Driven Development. Is foundation resides in model early requirements and models actors, goals and dependencies. Architectural design is always based on subsystems. A design language has been developed for this methodology called Tropos. Further reading is available by Castro and others [15, 14].

3 Knowledge Management

Nowadays, we read the word Knowledge Management in nearly every important business magazine. That is because, Knowledge Management has changed the way that

enterprises think. Knowledge Management has introduced a whole new paradigm about the knowledge that the enterprise has and how to convert that knowledge into value.

Definition of Knowledge Management There is no strict definition of knowledge, hence the not so accurate definition of Knowledge Management. But, for this case Knowledge Management stands for the way in which companies generate value through intangible assets. These assets could be encapsulated into the term called knowledge, which should not be mistaken with data or information. Also, managing something intangible like knowledge proves to be not a simple task. Many companies have adopted the Knowledge Management approach, even they have created a position called Chief Knowledge Officer.

In the case of knowledge, there is a small classification. The first one is tacit knowledge, tacit means that is still inside the brain or in the vessel of knowledge (the body). And the second one is the explicit knowledge, is the one that is already available for deployment or distribution in a digital form.

In the other hand, the use of knowledge has generated a continuous improvement cycle in nearly every company that has used it. Mostly because one of the key characteristics is its versatility and dynamic process, in other words, it never stays the same. It would be rare to see a strategy based on Knowledge Management that has been used for ten years straight. Another important key aspect of Knowledge Management is that Knowledge Management is always hungry, by hungry it is meant that more knowledge needs to be generated in the course of time. And with this new knowledge comes the need to apply it to generate value, and thus have a better, wiser and healthier company.

It is important to mention, that Knowledge Management does not only consist on reorganizing the key processes in the company to add value through intangible assets. It consists in generating strategies, managing intangible assets, creation and innovation, creating a knowledge based economy, developing cities that use knowledge as their main source of income, and many other disciplines that revolve around the Knowledge Management. This disciplines focus in harnessing the power of knowledge, and believe

Knowledge Management	*Business Intelligence*
1. Capture data	1. Capture data
2. Organize data	2. Organize data
3. Analyze data	3. Analyze data
4. Aggregate data	4. Aggregate data
5. Apply data	5. Apply data
6. Create new knowledge	6. No equivalent action!
7. Knowledge dispersion	7. No equivalent action!

Table 1: Differences between Knowledge Management [20].

it or not "there is no knowledge that is not power".

Needless to say, one of the most important aspects of Knowledge Management it's the ability to distribute the knowledge in a timely fashion. When anyone requests information, it is expected to be quick and true. With de distribution of knowledge it happens the same way, companies want more knowledge that comes with more value, which comes with a greater process which gives the company more money (quite simple knowledge -> value -> better process -> more profits).

Another important point is that Knowledge Management, should not be mistaken with Business Intelligence. The main difference between this "strategic siblings" is that Knowledge Management has the task to generate new knowledge , while BI does not support this important step. And this step is what makes more productive the use of Knowledge Management over BI. Note that Knowledge Management and BI are suited for two separated purposes, and as it was stated above both of them should not be mistaken.

3.1 History of Knowledge Management

The mid 1980's, the need to harness the power of knowledge arises. But it's until the early 1990's, when companies start to exploit the information or knowledge that every worker has. With time, the Knowledge Management scene changed and adapted several models. Nowadays, Knowledge Management is in every big company. To prove it we can take the example of Cemex, thanks to Knowledge Management this Mexican

enterprise has grown to be one of the most important cement producer in various countries. And not only in Mexico, there are several countries that have gambled to the Knowledge Management approach. These countries are looking at the results right now.

One of the basic needs that Knowledge Management solves is "the failure of financial models to represent the dynamics of knowledge", by proposing a more flexible method that can cover all the movements and flows of knowledge. Through time Knowledge Management has been reinventing itself, by adding more perspectives. One of the latest perspectives that has adopted is the use of IT for the distribution of the knowledge that is generated. While IT has become an important part of Knowledge Management, it is important to note that IT is not Knowledge Management, and vice versa.

3.2 ¿What is an intangible asset?

An intangible asset is formed by the key information that can produce value to the company, then it is important for a company to classify al the information that is going to be used as knowledge. Also, intangible assets cannot be measured in quantity or in terms of money. Instead, a good appreciation of intangible assets can be gained by the tools being used to exploit that asset.

Remember that intangible assets are also called by various names, most of them use the attribute of intangibility. Some of them are intellectual capital, best practices, identification of sources of knowledge, strategies based in knowledge. In other words, every intangible asset that cannot be touched and is deeply rooted within the great tree of knowledge. Thus, we reach the conclusion that an intangible asset is the same as intellectual capital.

Intellectual capital as we know it has its own subset of little capitals that give form to it, this scheme has been debated through the years. The following structure was defined a while ago, this structure manages three important parts of the intellectual capital [19].

- Organizational capital: In this capital the brand, de positioning, mission, and other marketing & operational assets of the company are present.
- Human capital: In this area, the knowledge of the workers (or the knowledge workers) can be found in here. Also, the core competencies can be found in here. This because the core competencies are created by the workers that use knowledge to add value to its final product.
- Relational capital: This part refers, to the quality of the relationship with other agents that play a vital role in the processes of the company. The greater the quality, the greater the knowledge is.

3.3 Objectives of Knowledge Management

As stated by Karl M. Wiig the chairman of the Knowledge Research Institute: "the objectives of Knowledge Management are: to make enterprises act intelligently as possible to secure its viability and overall success and to otherwise realize the best value of its assets" [31]. With these objectives in mind, a model can be created to suit every one of the companies that area trying to apply principles of Knowledge Management.

The objectives of Knowledge Management, are focused towards the exploitation of the knowledge that every worker in a company has. Although this knowledge has to be filtered, to ensure that the knowledge that is being used has the necessary components or richness to produce value and aid a company in its processes. From this idea the concept knowledge worker is extracted, a knowledge worker is a simple worker that can improve his work by using the knowledge that he has.

Bear in mind the fact that Knowledge Management is always moving, and so does the scope of their objectives. The objectives stated above make us believe that the company can have an own mind, a concept that is not too far away from the reality. And all the knowledge that is generated by the company must be managed, and there is where Knowledge Management comes into action.

3.4 Tools and aids

There have been many tools that Knowledge Management has created. They vary from online portals that assure the retention of knowledge, strategies based on the intellectual capital that the company has, and many others that have been mentioned. The main purpose of the tools that Knowledge Management has created is to retain the knowledge of the best workers and to document the best practices of every department. These tools have evolved through time, the best example of this evolution are the well known best practice manuals. These manuals, have been around ever since the creation of the concept of a company and their operation manuals. The thing that differentiates a best practice manual from an operation manual, is the amount of knowledge that has been used to refresh the content of the manual. These tools have proven to be a useful ally when it comes to creating competencies. Creating competencies (or processes that differentiate a company from others in the same market) is one of the many "hidden" objectives that Knowledge Management has. Figure 6 shows the relations between tools and techniques.

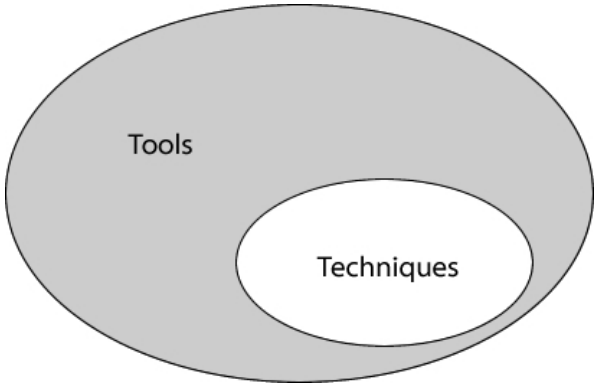

Figure 6: Relation between tools and techniques.

It is important to have in mind, the fact that there are techniques and tools inside Knowledge Management. A technique gives the ability to manipulate, discover and generate new knowledge. A tool deploys and distributes the knowledge that has been generated by a technique. So, every technique is a tool, but not all of the tools are

techniques. In the next section a set of tools and techniques of Knowledge Management with their description are presented [16]:

Business Intelligence tools: As it was stated earlier, the principal function of this type of tools is to visualize large amounts of data. This data is analyzed and can bring important results, when properly interpreted. An example, Microstrategy 7i , is one of the most well known BI suites.

Content Management System (CMS): A tool that enables the distribution of media contents through a web based platform, with some rules and restrictions of content. One of the modules in Hummingbird Enterprise 2004 manages the aspect of CMS.

Data analysis This technique generates new knowledge through the data that is generated from inside the company.

Electronic Document Management: Documents are the source of knowledge, so it must be spread. This type of tools allows the organization and deployment of key documents in a Knowledge Management based approach. One clear example of Electronic Document Management is Docuware [26] pioneers in electronic filing.

Knowledge map: K-maps represent the flow of knowledge through a company. One of the most prominent examples of this tool can be found within Innoval's K-map software.

Groupware: This tool acts as a platform for collaboration and communication between persons in a company. It also includes Electronic Document Management and other tools. It is important to mention that Groupware applications have taken the attention of various programmers, and many open source tools have been created. The most prominent one is phpGroupware.

Agent technologies: This tool mingles with Artificial Intelligence and creates a customized search engine based on a personal profile. In this case knowledge comes packed in a customized way. A company called Agivity is proposing this type of tool, on their site they have information and applications of this kind of tool.

Case-based reasoning: This technique allows the knowledge to be subtracted from a case and then to be applied to a similar situation.

Skill management: This tool documents and maintains the skills that employee has. And not only skills, also competencies which later allow the knowledge manager to create an expert directory. ExecuTrack a German company offers ETWeb as a solution for "talent managing" and "talent planning".

Abstracting This technique allows the creation of summaries and abstracts. This technique is widely used to capture a reader's attention.

Creativity support This type of tools support the creative process in a product or innovation stage. Commonly called brainstorming support. One of the classic tools in this section is Microsoft's Visio [6].

Knowledge discovery: This technique is the one that differentiates BI from Knowledge Management. Using techniques and tools like data mining and data warehousing, knowledge is created from the interpretation of this data.

Categorizations: Use of semantic and logic analysis algorithms to sort texts into a more organized and categorized document repository.

Computer based training: Tools that support Just In Time training for employees, where knowledge can be obtained or even new sources of knowledge can be created.

Workflow management system: If the processes within a company are the mayor source of knowledge, then these processes must be monitored to gather knowledge. Workflow Management System manage the communication and the information that

exists between processes. This type of tool is integrated in most Knowledge Management suites.

4 A Knowledge Management Framework for Software Engineering

Some of the tools that where mentioned above are part of the KM tools that have been created, these tools are being implemented in various companies around the globe. No matter what type of activity the company does, whether it's producing carbonated drinks or manufacturing steel blades for razors. The framework that is presented in this section comprises several tools of KM and their application to the diverse development stages and in general [13].

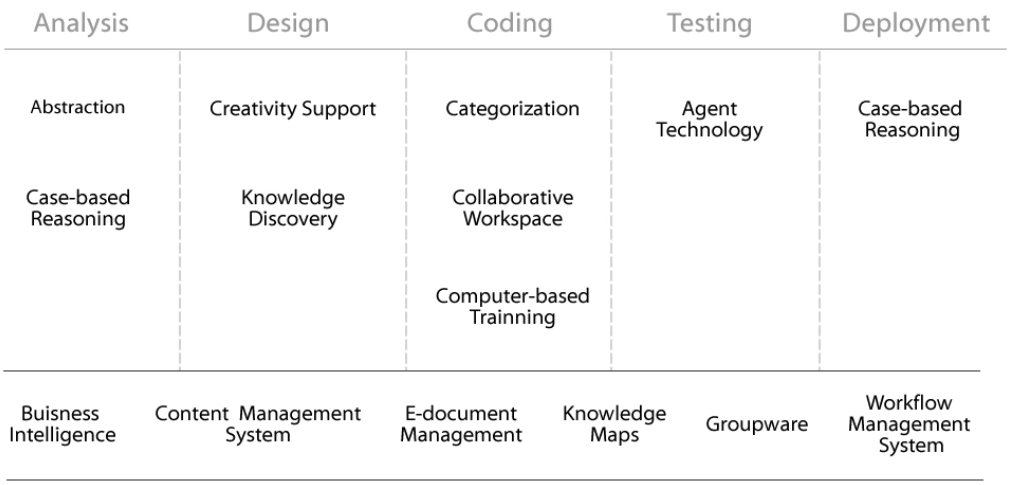

Figure 7: Knowledge Management Techniques distributed in the development lifecycle

4.1 General tools

There are a set of tools that KM offers that can be applied to any of the development stages that SE has, even planning can be included. Every each of them has a reason to be, and a reason to be there. The tools that apply through the whole SDP are:

exists between processes. This type of tool is integrated in most Knowledge Management suites.

4 A Knowledge Management Framework for Software Engineering

Some of the tools that where mentioned above are part of the KM tools that have been created, these tools are being implemented in various companies around the globe. No matter what type of activity the company does, whether it's producing carbonated drinks or manufacturing steel blades for razors. The framework that is presented in this section comprises several tools of KM and their application to the diverse development stages and in general [13].

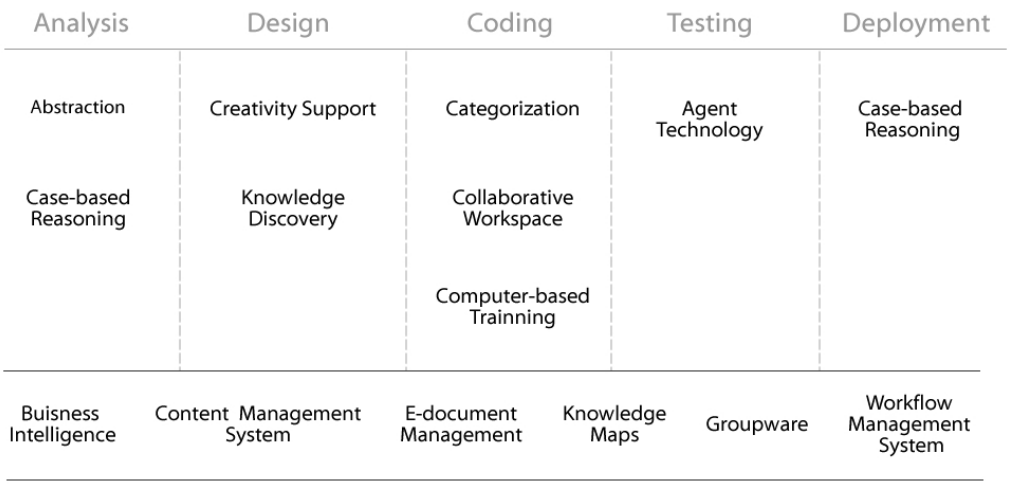

Figure 7: Knowledge Management Techniques distributed in the development lifecycle

4.1 General tools

There are a set of tools that KM offers that can be applied to any of the development stages that SE has, even planning can be included. Every each of them has a reason to be, and a reason to be there. The tools that apply through the whole SDP are:

4.1.1 Business Intelligence

By creating and analyzing data, the companies that develop software can take advantage of the great amount on information and data that is generated in this kind of companies. Also, it would be a lot more natural to have data mining processes in companies that develop software.

4.1.2 Content Management System

Through the develop process, a decent amount documents are used mostly on the testing phase. With a CMS all of those documents can be deployed and made available for use whenever, wherever is needed. For example, when the testing annotations are published, immediately the programming staff can correct errors and learn from the errors that were made.

4.1.3 E-document Management

With the use of the Electronic Document Management, the whole programming staff can have a repository of documents that range from complaints of clients to technical documents where requirements that are similar to the ones that the client is giving.

4.1.4 K-maps

To exploit the full potential of knowledge, the flow or the "dynamics" of knowledge must be mapped. Imagine a software developing process that has an input of knowledge and an output of knowledge assets, this example can be reached within the use of K-maps. These maps can show the flows of knowledge between processes or even between members of the same team.

4.1.5 Groupware

By having a space were documents can be shared, the knowledge information and creation can be dramatically improved. In this case, every member of the project that is been designed, coded or tested can work in different locations of the world on the same module without being in the same physical location. Another advantage of using

groupware is that has become one of the cheapest KM tools, hence the great amount of open source projects that are generating groupware applications.

4.1.6 Workflow Management System

With the aid of a WMS, can generate quality communication links that not only generate value in the process also it can improve the way that teams perform. And automating the workflow based on the knowledge (consumer or internally generated) proves to be one of the most efficient (financially speaking) techniques more than cost reduction.

4.2 Tools for Analysis

One of the most critical aspects of the SDP, is the analysis. If requirements are generated without the proper techniques, it is sure that the project will fail. In this part 2 techniques are used, both techniques use texts as their primary source of knowledge. They are presented in the next section.

4.2.1 Abstraction

By creating a summary of all the requirements, or the needs of the client. The analysts can improve the way they generate the requirements for the programming team. Other advantage can be achieved by creating a summary of the texts that are given as optional reading for the analysts, with this practice the methods of generating requirements can be improved with each project.

4.2.2 Case-based reasoning

Using cases as a "crash dummy" for the analysis phase can be helpful when dealing with big companies or elaborated projects. Analysts can also generate knowledge on various situations and become prepared for any situation.

4.3 Tools for Design

The process that involves more creativity is the designing phase, thus the need for knowledge tools and techniques that support the creative process. The tool that fits

this process is Creativity Support, while the technique is Knowledge discovery. Both of them are explained below:

4.3.1 Creativity Support

This tool implies that creativity is exploited to its fullest. The design phase is the one that can receive the benefits of having a more creative team of designers, and with the added purpose of adding value and generating knowledge for further projects. The creativity support systems can dramatically improve the quality of a project, even tap into new sources of creativity.

4.3.2 Knowledge Discovery

After a design has been made, the feedback that receives can be turned into knowledge by being documented and later distributed. With this knowledge, further designs can be made in less time so there is more time for feedback.

4.4 Tools for Coding

This phase is the one that takes the most time. This phase requires a great amount of work, but this amount of work can be done in a better war applying some of the KM tools and techniques. In the next section the tools and techniques that fall into the coding phase are explained.

4.4.1 Categorization

With this technique, the documents or data referring directly the coding process and its best practices can be stored for later re use. Also it generates a source of knowledge where the team members of a project can later on, check those documents for a more specific knowledge based approach. This technique involves further processes in the Software Development Lifecycle.

4.4.2 Collaborative workspace

The purpose of this tools is having a place where all the programmers can join and work together in a single module. But at the same time, it works as an administrator of several teams that are geographically separated from each other. It breaks the paradigms of an office work.

4.4.3 Computer based training

By having a training based on knowledge, the trainees will become more competent on their own. This kind of training is feed by the knowledge, this knowledge is generated by various fonts and sources. The training leads programmers into a continuous skill improvement cycle, were the newest trends are adapted.

4.5 Tools for Deployment

The cases are based on real situations and represent an opportunity to improve the overall performance in this aspect. The cases can be later solved an distributed through the whole organization so all of the team members come to an equal solution.

4.6 Benefits towards a new level on Capability Maturity Model

The first benefit that Knowledge Management will bring to Software Engineering will be the maturity increase of the process. Software Engineering Process is how decisions are taken in which methodology to use, which tools for several projects in a software production team. Even tough a process differs from team to team. Is been stated that Software Engineering Processes can be review through the Capability Maturity Model (CMM).

Paulk [22] establish five levels: Initial, Repeatable, Defined, Managed and Optimizing. An empirical finding in this research implies implementing tools by stage and state general tools as a team sees good to go forward.

.

5 Conclusion

Knowledge Management has a breath application in Software Engineering. Even tough Knowledge Management With the application of this framework, innovation will be exploited in Information Technology departments for increased performance in terms of time and budget.

Further steps will include experiment and refinement of the framework presented in this paper. Analysis of it from both fields at Tecnológico de Monterrey will give a complete feedback in the process. Future work includes the results of basic experiments of this framework.

Glossary

A

Analyst A person who analyzes or who is skilled in analysis in a specific domain field. He gathers requirements to transform them in specifications of what a software system should do., p. 3.

C

Client In general, someone or something receiving a service of some kind. Within computing the term frequently refers to one element of a client/server system, typically an application , that communicates with the end-user by means of a server [24]., p. 4.

D

Database designers Database designers take in charge of planning, deployment and maintenance of databases used in software to store data., p. 8.

I

Information Technology The study or use of systems (especially computers and telecommunications) for storing, retrieving, and sending information [24]., p. 1.

J

Java A programming language which was developed by Sun Microsystems. It has a number of features which make it an excellent medium for developing programs for the Internet. The first is the fact that it contains a large number of facilities for carrying out tasks such as connecting to another computer on the Internet and sending data to it [24]., p. 8.

K

Knowledge Management Knowledge Management (KM) is a mature trend for the new wave of enterprise innovation. It permits to storage the intangible assets of a company to standardize processes and to potentialize improvements for better performance., p. 1.

O

Object Management Group The Object Management Group (OMG) is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications [1]., p. 8.

Q

Quality Assurance The process of ensuring that a software system and its associated documentation are in all respects of sufficient quality for their purpose. While a quality assurance team may be involved in all stages of a development project, there is typically a recognized quality assurance activity following completion of development and prior to release of the system for operational use.[24], p. 5.

R

Requirement What a proposed system must be capable of doing to solve the problems of a defined set of potential users of such a system [24].

S

Software Engineering Software Engineering (SE) is the science for develop today's complex computer programs. It provide diverse methodologies to fully accomplish projects in terms of requirements, time or budget., p. 1.

Software Specification A document that defines what a program or software system is required to do and the constraints under which this required functionality must be provided. These constraints are often referred to as nonfunctional requirements; they may affect the way in which the software is developed (e.g. for safety-critical or security-critical software) or may impose physical limits on the space, size, and performance of the software to be developed [24]., p. 3.

Stakeholder Anybody with some form of interest in a business. As well as shareholders , this includes directors, managers, other employees, customers, subcontractors, and even the general public in cases where the firm's activities impact on the environment [24]., p. 4.

References

[1] Object management group overview. <http://www.omg.org/gettingstarted/gettingstartedindex.htm>, last visited December 11th 2004.

[2] Beck, K. Extreme programming. In *TOOLS '99: Proceedings of the Technology of Object-Oriented Languages and Systems* (1999), IEEE Computer Society, p. 411. IEEEXplore ARN 779100.

[3] Boehm, B. A spiral model of software development and enhancement. *Computer 21*, 5 (May 1988), 61–72. IEEE Xplore ARN 59, DOI 10.1109/2.59.

[4] Budgen, D. Software design: An introduction. In *Software Engineering*, R. Thayer and M. Dorfman, Eds., 2nd ed., vol. 1: The Development Process. IEEE Computer Society, Los Alamitos CA, USA, 2002, ch. 5, pp. 197–208. LC Classification QA76.758 .S64 2002 v.1.

[5] Christensen, M. Software construction: Implenting and testing the design. In *Software Engineering*, R. Thayer and M. Dorfman, Eds., 2nd ed., vol. 1: The Development Process. IEEE Computer Society, Los Alamitos CA, USA, 2002, ch. 7, pp. 377–410. LC Classification QA76.758 .S64 2002 v.1.

[6] Corp., M. Microsoft office visio overview. <http://www.microsoft.com/office/visio/prodinfo/default.mspx>, last visited December 11th, 2004, 2004.

[7] Coupaye, T., and Estublier, J. Foundations of enterprise software deployment. In *Proceedings of the Fourth European Software Maintenance and Reengineering* (Zurich Switzerland, March 2000), Dassault Systems & Imag Lsr Joint Laboratory, pp. 65–73. IEEEXplore ARN 827313, DOI 10.1109/CSMR.2000.827313.

[8] Dugan, J., et al. Hype cycle for application development, 2004. Strategic analysis report, Gartner Research Group, July 2004. Gartner Report ID G00120914.

[9] Elrad, T., Filman, R. E., and Bader, A. Aspect-oriented programming: Introduction. *Communications of the ACM 44*, 10 (2001), 29–32. ACM Number 383853, DOI 10.1145/383845.383853.

[10] Guevara, M. Global software development: A challenge for project management. *5° Certamen de Investigación Biblioteca Digital. Tecnológico de Monterrey.* (May 2004), 1–16. Colección de Documentos Tec DOTEC 118291.

[11] Highsmith, J., and Cockburn, A. Agile software development: The business of innovation. *Computer 34*, 9 (2001), 120–122. IEEEXplore ARN 947100, DOI 10.1109/2.947100.

[12] Holz, H., Melnik, G., and Schaaf, M. Knowledge management for distributed agile processes: Models, techniques, and infrastructure. In *WETICE '03: Proceedings of the Twelfth International Workshop on Enabling Technologies* (2003), IEEE Computer Society, p. 291. IEEEXplore ARN 1231423.

[13] I. Rus, M. L., and Sinha, S. Knowledge management in software engineering: A state-of-the-art-report. Unclassified report, Data & Analysis Center for Software, ITT Industries, 2001. <http://www.cebase.org/umd/dacs_reports/kmse_-_nicholls_final_edit_11-16-01.pdf>, last visited August 23rd, 2004.

[14] J. Castro, M. Kolp, J. M. A requirements-driven development methodology. In *CAiSE '01: Proceedings of the 13th International Conference on Advanced Information Systems Engineering* (2001), Springer-Verlag, pp. 108–123.

[15] J. Castro, M. Kolp, J. M. Tropos: A framework for requirements-driven software development. Department of Computer Science, Toronto University, <http://www.cs.toronto.edu/ mkolp/tropos1.pdf>, last visited November 20th, 2004, 2004.

[16] K. Martins, P. H. a. J. V., Ed. *Knowledge management : concepts and best practices*, 2 ed. Springer, Berlin, Germany, 2003. LC Classification HD30.2.K663 2003.

[17] Lanowitz, T. Lack of software engineering discipline can haunt you. Research note, Gartner Consulting Group, September 2003. Gartner Strategic Plan SPA-19-4351.

[18] Leach, R. *Introduction to Software Engineering*. CRC Press, Boca Raton FL, USA, 2000. LC Classification QA76.758 .L33 2000.

[19] Leal, P. El capital intelectual y su aplicación estratégica. Unpublished Draft, Centro de Sistemas de Conocimiento, Tecnológico de Monterrey., 2004.

[20] Nagy, D. The difference between business intelligence and knowledge management. Unpublished Draft, Center for Advanced Analytics and Business Intelligence.

[21] Papazoglou, M. P., and Georgakopoulos, D. Introduction: Service-oriented computing. *Communications of the ACM 46*, 10 (2003), 24–28. ACM Number 944217, DOI 10.1145/944217.944233.

[22] Paulk, M. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison Wesley, Reading, MS, USA, 1995.

[23] Pokkunuri, B. Object oriented programming. *SIGPLAN Not. 24*, 11 (1989), 96–101. DOI 10.1145/71605.71612.

[24] Press, O. U., Ed. *A dictionary of Computing*. Oxford Reference Online, 1996. Digital Library: Oxford Reference Online.

[25] Pressman, R. Software engineering. In *Software Engineering*, R. Thayer and M. Dorfman, Eds., 2nd ed., vol. 1: The Development Process. IEEE Computer Society, Los Alamitos CA, USA, 2002, ch. 1, pp. 9–26. LC Classification QA76.758 .S64 2002 v.1.

[26] PressWire, M. Aurora: New version of docuware document management system from aurora. *M2 Communications* (October 2003), 1. Proquest DID 434853111.

[27] Radding, A. Extreme agile programming. *Computerworld 36*, 6 (February 2002), 42–44. Proquest DID 105425893.

[28] Rising, L., and Janoff, N. The scrum software development process for small teams. *IEEE Softw. 17*, 4 (2000), 26–32. IEEEXplore ARN 854065, DOI 10.1109/52.854065.

[29] Sommerville, I. *Software engineering*, 6th ed. Addison Wesley, London, 2001. LC Classification QA76.758 .S657 2001.

[30] Studios, J. Uml class diagram for a java system. Electronic Image, 2004. <http://www.juicystudio.com/tutorial/java/images/object.gif>, last visited October 11th, 2004.

[31] Wiig, K. Knowledge management: An introduction and perspective. *Journal of Knowledge Management 1*, 1 (1997), 6–14. Emerald Database ID 13673270/v1n1/s1.

[32] Yingxu, W., and King, G. *Software engineering processes*. CRC Press, Boca Raton FL, USA, 2000. LC Classification QA76.758 .W38 2000.