UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Gestionale

SOFTWARE PEOPLE AND SOFTWARE QUALITY: A QUALITATIVE, EXPLORATORY RESEARCH ON HOW WORKFORCE MANAGEMENT CAN ENHANCE THE PRODUCT QUALITY IN SOFTWARE COMPANIES.

RISORSE UMANE E QUALITÀ DEL SOFTWARE: UN'INDAGINE EMPIRICA.

Relatore: CH.MO PROF. ETTORE BOLISANI

Laureanda: ELENA SORTINO

ANNO ACCADEMICO 2010 – 2011

Abstract

The aim of this work is to investigate how to address the management of individuals in a software company to positively affect the quality of the software product. The basic idea is that the better people are managed, the better is the quality of the product and, consequently, the more the customer is satisfied.

This paper goes across the literature to offer an overview on some main aspects of the management of knowledge: reasons for managing knowledge, effective strategies, adopted methods. The management of a software company is also examined: models of software development processes are described, methodologies for processes' assessment and to guide improvements are presented, approaches to people management are deepened. Particular attention is drawn to software quality, how it can be defined and measured and its relationships with both the development process and the user/customer perspective (in terms of needs and satisfaction).

Within the literature very few are the explicit references to the relationship between software quality and software people, because of the opposite attitudes towards this concept: it's something taken for granted or else not considered at all. For this reason, a questionnaire has been designed to collect information on this subject. The questionnaire has been submitted to several software companies and the answers have been analyzed. Furthermore, interviews to managers within the software development area have been arranged to gather more exhaustive information .

Consequently, this work provides a description of how individual competencies, team skills and people's involvement in software development are managed in some real cases. Moreover, this project shows how much companies actually understand that effectively managing their people affects the quality of their software products. Conclusions on how software companies may improve their people management in order to enhance the quality of their software are drawn.

Abstract

La tesi analizza l'effetto della gestione delle risorse umane sulla qualità del prodotto nelle società di software. In particolare si ipotizza e si cerca di verificare se a una più efficace gestione del capitale umano (e in particolare lo sviluppo e l'applicazione delle conoscenze/competenze del personale) corrisponda una migliore qualità del software, da cui una maggior soddisfazione del cliente.

Dopo aver approfondito, nella letteratura sul knowledge management, i presupposti e gli approcci alla gestione della conoscenza nelle organizzazioni, vengono esaminati i tipici modelli di gestione di una società di software e di sviluppo del prodotto software. Particolare enfasi viene posta sul problema della qualità del software e della sua misura, anche in relazione ai bisogni del cliente.

Constatato che in letteratura sono ancora insufficienti gli studi che pongono in relazione la qualità del software con le competenze/conoscenze delle risorse umane e la loro gestione, si è quindi impostata una ricerca sul campo basata su un questionario presentato ai manager di varie imprese di software in Svezia e Italia. A complemento di queste informazioni sono state condotte interviste mirate ad alcuni manager dell'area sviluppo software in tali imprese.

Lo studio fornisce una descrizione di come le conoscenze/competenze individuali e di gruppo sono gestite nello sviluppo del software in riferimento ad alcuni casi reali e mette in evidenza l'effettivo livello di comprensione di come ciò influenzi la qualità del prodotto. Vengono ricavate indicazioni su come le società di software possono migliorare la gestione delle risorse umane al fine di un miglioramento della qualità del software prodotto.

Acknowledgements

My thanks go to Vladimir Tarasov and to Ettore Bolisani for supervising and supporting this work.

I’d like also to thank those managers who have taken time to fill out the questionnaire. Special thanks go to Magnus Werner (SAAB Training and Simulation) and Anna Leo (Swedish Board of Agriculture) who took the time to answer my questions during the interviews. Without their collaboration this work would not have been possible.

I'm infinitely grateful to those people whose love and support have guided me every step of the way:

Mum and Dad, to whom I owe everything.

Chiara, whose strength inspires me beyond words.

Daniele, Veronica, Anna, Betty, Mauro, Lucia, who mean the world to me.

Margherita, Camilla, Luca, Michele, Ketty, Oberdan, Gabriella, Andrea, Laura, Giovanni, Orietta, Zezè, who have always been there for me as more than family.

Francesca, Antonio, Paola, Alessandro, Marco, Alice, Chiara, Alfredo, Fabio,
Filippo, Ambra, Giorgia, Giuseppe, Jacopo, Lucio, Michele,
Alberto, Umberto, all the people from ASU / Il Sindacato degli Studenti,
Luca, Monica, the people studying, teaching and working at the DTG
Department of Engineering and Management in Vicenza,
Fam. Furlan, Fam. Vanin, Fam. Folin, Raimondo, Fam. Zancan, Rita,
Sabbadin’s relatives, Guarnieri’s relatives, Emilia, Sr. Giuseppina,
Fabio, Ornella, Anna, Stefania, Francesca,
the Italian, Swedish and international students I've met in Jönköping,
Gunnel, Abram, Gustaf, my roommate Flavia,
and all my friends that make my life so beautiful.

Acknowledgements

I miei ringraziamenti vanno ai prof. Vladimir Tarasov e Ettore Bolisani per aver supervisionato e supportato questo lavoro.

Ringrazio anche i manager che hanno speso del tempo a compilare il questionario. Ringraziamenti speciali vanno a Magnus Werner (SAAB Training and Simulation) e Anna Leo (Swedish Board of Agriculture) per aver risposto alle mie domande durante le interviste. Senza la loro collaborazione questo lavoro non sarebbe stato possibile.

Sono infinitamente grata alle persone che con il loro amore e supporto mi sono state vicine ad ogni passo:

Mamma e Papà, a cui devo tutto.

Chiara, la cui forza mi ispira più di quanto le parole possano esprimere.

Daniele, Veronica, Anna, Betty, Mauro, Lucia, che per me sono tutto.

Margherita, Camilla, Luca, Michele, Ketty, Oberdan, Gabriella, Andrea, Laura, Giovanni, Orietta, Zezè, che ci sono sempre stati come e più di una famiglia.

Francesca, Antonio, Paola, Alessandro, Marco, Alice, Chiara, Alfredo, Fabio,
Filippo, Ambra, Giorgia, Giuseppe, Jacopo, Lucio, Michele,
Alberto, Umberto, l’ASU e Il Sindacato degli Studenti,
Luca, Monica, le persone che studiano, insegnano e lavorano al DTG,
Fam. Furlan, Fam. Vanin, Fam. Folin, Raimondo, Fam. Zancan, Rita,
parenti Sabbandin, parenti Guarnieri, Emilia, Sr. Giuseppina,
Fabio, Ornella, Anna, Stefania, Francesca,
gli studenti italiani, svedesi e dal mondo che ho incontrato a Jönköping,
Gunnel, Abram, Gustaf, la mia compagna di stanza Flavia,
e tutti i miei amici che rendono la mia vita così meravigliosa.

Key words

Software company, Knowledge Management, CommonKADS, Software development, P-CMM, Software quality.

Contents

List of Figures

List of Tables

List of Abbreviations

CEO: Chief Executive Officer

HR: Human Resources

IEC: International Electrotechnical Commission

ISO: International Organization for Standardization

IT: Information Technology

KM: Knowledge Management

KPA: Key Performance Indicators

P-CMM: People Capability Maturity Model

PSP: Personal Software Process

RUP: Rational Unified Process

SEI: Software Engineering Institute

SPICE: Software process Improvement and Capability Evaluation

SQA: Software Quality Assurance

SW-CMM: Software Capability Maturity Model

TQM: Total Quality Management

1 Introduction

In today’s economy the easiest way to increase profits seems to be cost reduction – especially for what concerns people management. This leads to temporary benefits that are destined to fade away, leaving companies with nothing more than people without any sense of belonging, whose competencies are felt to be constantly underestimated.

A different approach to business growth, based on people, can be undertaken: the basic idea is that the better people are managed, the better is the quality of the product, the more the customer is satisfied and the sales are likely to increase.

This chapter sets the purpose and objectives of the study and describes the issue that the paper aims to deepen. A short background of the project is given and limitations of the project are pointed out. Finally an outline of the report provides an overview of the remaining chapters.

1.1 Background

Managing a software company means to manage its people, processes and products. Even if in reality each of these aspects may be managed by a different organizational function, they shouldn’t be considered detached. Fig. 1.1 highlights these connections, as software developers inherently belong to processes that lead to the production of software.

Knowledge management is the branch of management which deals with individual competencies and information exchanges. It provides practices in order to identify, collect, represent and distribute knowledge, both on individual and organizational level. As nowadays knowledge is the most important resource for organizations, a competetive advantage comes from knowledge management.

In order to manage a software company, its processes must be identified, assessed and continuously improved, since effective processes positively affect the quality of the product. Processes in a software company have a distinctive characteristic: people play a basic role for product realization. So, the way software developers take part in the processes needs to be assessed and improved as well.

Managing a product such as software means managing its quality. Quality can be approached from several different perspectives.

Chapter 2, 3 and 4 provide the theoretical background to this project. The following themes are approached: the meaning of knowledge and of its management; strategies, methods and techniques to promote an effective knowledge management; software processes, software people, their management and assessment; software quality, its definition and measurement.

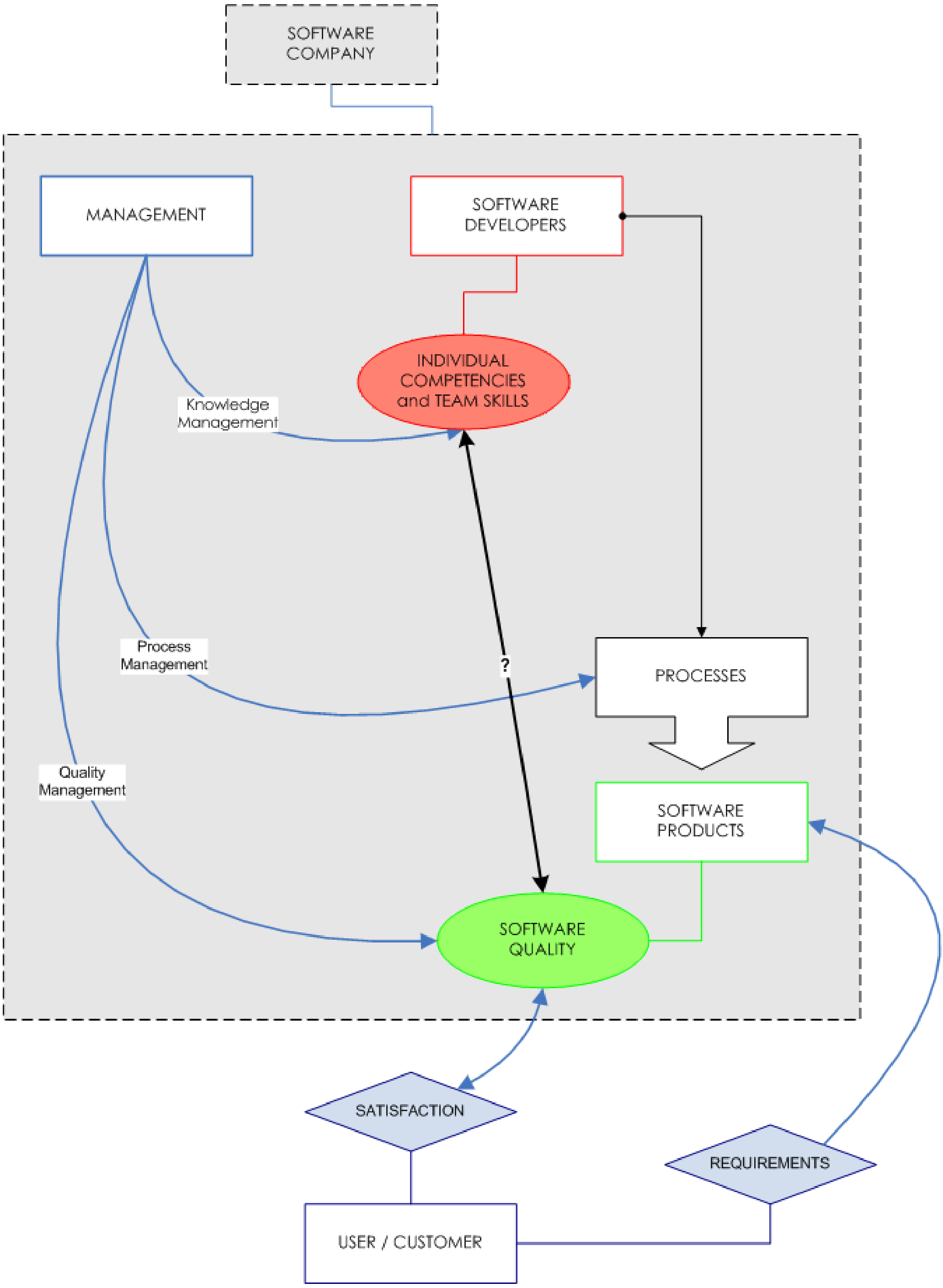


Fig. 1.1 Aspects to be considered while managing a software company

1.2 Purpose/Objectives

The purpose of this work is to investigate whether there is a relationship between software people and software quality (see Fig. 1.1), so that a better management of the people can improve the quality of the product.

Within the literature very few are the explicit references to the relationship between software quality and software people. Therefore, a confirmation from the reality of software companies is the objective to be pursued.

In order to achieve this goal, the following research questions should be answered:

- How do software companies manage individual competencies?
- How do software companies manage quality?
- How much software companies actually understand that effectively managing competencies can improve the quality of their products?
- How do software companies manage the relationship people – quality?

1.3 Limitations

This study points out that a particular relationship, the one between software people and software quality, it’s worth being investigated. The project doesn’t aim to be exhaustive: on the opposite, it strives for being a starting point.

The project is qualitative rather than quantitative, both because of the lack of time and resources and because of the novelty of this perspective of study.

Results and conclusions cannot be generalized, since the study concerns just few companies, a very little part of reality. Once again, the objective is to underline the importance of further investigation rather than drawing global conclusions.

However, this study can still provide some concrete hints, as it depicts pieces of reality that can be immediate source of inspiration.

1.4 Thesis outline

The thesis is divided into seven main chapters.

The literature review is the theoretical background of the project and takes Chapter 2, 3 and 4. Each chapter elaborates on one of the three main themes identified as the main activities for the management of a software company: knowledge management (Chapter 2), processes and people management (Chapter 3), and software quality management (Chapter 4).

Chapter 5 describes the research strategy underlying this project and the choice made: the theoretical approach, the data collection, the analysis of the results and the way conclusions are drawn.

In Chapter 6 and 7 the results of the investigation, made through questionnaire and interviews, are presented and discussed.

2 An Overview on Knowledge Management

2.1 What is knowledge and how can it be managed?

2.1.1 The nowadays "raw material": knowledge

A brief history

Despite knowledge management is nothing new for mankind, it's only since 1990s that it has become a conscious practice within companies. As Hansen, Nohria and Tierney [22] point out, the shifting of the focus from natural resources to intellectual assets and the rise of networked computers have made possible for managers to investigate what knowledge underlies their businesses and how that knowledge is codified, stored, shared and used.

Knowledge is the most important resource for organizations, the nowadays principal "raw material". As Schreiber et al. [42] recall, writers on management estimate that intellectual capital constitutes for 75% to 80% of the total balance sheet of companies. Thus, managing knowledge is a crucial activity in organizations.

Defining knowledge

What is knowledge? To answer to this question we can resort to the rigorous although ordinary definitions of data, information and knowledge. Knowledge is the whole of data and information that people use to carry out tasks and create new information. By data we mean the uninterpreted signals that constantly reach our senses; information is data with a meaning.

However, Schreiber et al. [43] suggest that a rigorous answer to what knowledge is doesn't really matter because everyday we can easily recognize which are the people that embody knowledge and what knowledge is used within certain actions.

An observation should be made: knowledge depends on context. That's why data, information and knowledge are defined depending on the situation: for example, in knowledge engineering knowledge is considered task- and domain-specific.

Furthermore, knowledge can be detected at different levels: individual, group or organizational level [20].

Tacit vs explicit knowledge

Knowledge can be tacit or explicit. Tacit knowledge, also referred to as knowing, is the deeply rooted know-how that emerges from action in a particular context. Explicit knowledge on the other hand is the so-called know-what that can be extracted from the knowledge owner and shared with other individuals.

How knowledge is created is described in “The Knowledge-Creating Company”, a book of Nonaka and Takeuchi [46]. Four modes of knowledge production are identified:

- Socialization: from tacit to tacit knowledge, through showing something rather than speaking
- Externalization: from tacit to explicit knowledge, through writing practices and procedures
- Combination: from explicit to explicit knowledge, through the integration of pieces of knowledge
- Internalization: from explicit to tacit knowledge, through learning from repeating the same task many times.

According to these authors, all the four types of knowledge production take place within the organizational knowledge creation. Knowledge management should support these processes.

2.1.2 Managing knowledge: Why? How?

What drives the knowledge-management efforts?

Knowledge management comprises a wide range of practices to identify, collect, represent and distribute knowledge, both on individual and organizational level.

The efforts required, typically focused on organizational objectives such as improved performance, competitive advantage, innovation and sharing of lessons learned, are driven by some motivations [76]:

- making available more knowledge within products and services development
- reducing lead times

- facilitating organizational learning
- leveraging expertise across the organization
- increasing network connectivity between internal and external individuals
- solving problems
- managing intellectual capital and assets in the workforce.

Starved of knowledge

Although the progress in the field of information technology goes on and the size of information that people can collect is bigger and the process becomes faster day after day, it looks like "we are drowning in a sea of information, and starved of knowledge", as the futurist John Naisbitt said [30].

As Junnarkar [30] points out, what is still missing is the ability to make sense out of all this information. Improvement can go by two ways: enhancing the individual sense-making ability or leveraging the collective intellect of the people within the organization.

Therefore, knowledge management can be seen as the task of managing the way people connect with information (value is created, an advancement of understanding takes place) and with other people (i.e. how they relate to each other).

Junnarkar [30] underlines how it's natural for people to try to collect as much information as they can before trying to make sense out of it as long as looking for information is simpler than analyzing it. However, making sense out of the available information is much quicker than accumulating and analyzing every single existing piece of information: that's why knowledgeable people, the ones who can make sense out of not-necessarily-complete information, are required.

Therefore a competitive advantage comes from the matching between incomplete but timely information and people who can make sense out of it. Nevertheless, people tend to move towards a completeness of information instead of clarity of understanding; IT systems push them in this way too.

Connecting people isn't enough to leverage their collective intellect: they need to actually relate to each other. Naturally people tend to relate to each other thanks to the nature of their work, their work location and, very often, the values they share. Junnarkar [30] highlights that if there is a relation between people, a knowledge community can arise. So, the first enabler for learning and sharing is the human network, not the IT network.

Reporting from Wiig, Junnarkar [30] defines the process of knowledge management as the process of integrating information (collecting, codifying and organizing), making sense out of it and ensuring its continuity. The steps that should be undertaken include integrating information from both internal and external sources, enabling people-to-people and people-to-information connections and exploiting every way of making connections.

Junnarkar's methodology for knowledge management

Junnarkar [30] has developed a methodology for knowledge management. The aim of his methodology is to elaborate on the multitude of dimensions that define knowledge management, such as organizational strategy, information, use of IT, people and both individual and collective sense making, organizational culture, learning and sharing amongst individuals, measurements and so on.

The methodology relies on the following:

- Learning map: a visual description of the company's business model (markets, customers, competitions faced, distribution outlets, etc.) that aims to help people make sense of the business model and of some extent of the strategy .

- Values map: a description of the core values of the corporate culture, but also of teams values profiles, so that team members are aligned on the same values and knowledge communities can arise.

- Information map: a two dimensions matrix, location of the information (internal or external) – information's nature (quantitative/structured or qualitative/unstructured), that shows the different types and sources of information the company deals with.

- Knowledge map: analysis of how individuals learn and create knowledge and of how teams create insight. To explain the dynamics of individual knowledge creation and the teams' interactions over knowledge Junnarkar refers to Nonaka and Takeuchi's framework described before.

- Measurements: the balance scorecard proposed by Kaplan and Norton is used to track progress of knowledge management initiatives.

- IT map: a picture of how IT is used to enable the knowledge-management processes must be prepared after the first four maps have been completed.

Human roles within knowledge communities

As previously mentioned, knowledge communities rely on human networks rather than on IT. Humans can play different roles:

- steward/shepherd, who ensures that connections within the team takes place

- project/team leader, who manages the team's efforts

- cross-pollinator, who moves between teams and transfers knowledge

- knowledge team, which indexes and catalogues information needed by various teams and is in charge of the knowledge catalogue (a database of sources of information)

- topic experts, who have specific competences in various areas, are the ones to whom people turn to for having their questions answered and are the sense makers within the organization.

As Junnarkar [30] concludes in his article "the true leverage for any organization lies in the collective intellect of its people". Even if the role of IT is increasingly important, people's ability to make connections cannot be replaced. That's why knowledge management should foster the correct interaction between people, information and IT.

2.2 Strategies and organizational culture for an effective knowledge management

2.2.1 Computer-based vs people-based strategies

Rely on computers or on people?

Examining consultant firms, Hansen, Nohria and Tierney [22] have found that there isn’t a unique approach to managing knowledge. In some companies the strategy centers on computers and knowledge is codified and stored in databases – it’s the so-called codification strategy – while in others the strategy relies on people and knowledge is shared through direct person-to-person contacts – and goes by the name of personalization strategy.

Hansen, Nohria and Tierney [22] have analyzed computer companies and health care providers as well: they have found the same two strategies at work, leading to the conclusion that “the choice between codification and personalization is the central one facing virtually all companies in the area of knowledge management”.

Codification vs personalization strategy

Knowledge management strategies can be classified according to two main classes: codification strategies and personalization strategies.

A codification strategy uses a people-to-documents approach. It’s based on the assumption that knowledge can be extracted and codified from the person who developed it, so that any other person can search for it without having to contact the original owner.

By contrast, a personalization strategy focuses on dialogue. It relies on the knowledge sharing through interpersonal communication: knowledge is transferred through a connection between individuals (brainstorming sessions, one-on-one conversations, telephone, e-mail, job rotation), so that a network of people is created.

2.2.2 A strategy (perhaps the right one!) should be chosen…

“Do not straddle”

In all the companies Hansen, Nohria and Tierney [22] have examined, a distinct knowledge management strategy was chosen: even if both the approaches are used, they are not used to an equal degree.

Hansen, Nohria and Tierney [22] give an explicit advice, “do not straddle”: to effectively use knowledge one strategy should be predominantly pursued and the other one should be a support.

They assess at 80% the knowledge that should follow one strategy and at 20% the other.

How to choose the right strategy

Hansen, Nohria and Tierney [22] address some questions to managers in order to help them in recognizing the right strategy:

- “Do you offer standardized or customized products?”
- “Do you have a mature or innovative product?”
- “Do your people rely on explicit or tacit knowledge to solve problems?”

A personalization strategy should be considered when a customized product is offered; innovation is also best supported. A person-to-person approach also works best for people using tacit knowledge.

On the other hand, a codification approach best suits standardized, mature products and, of course, the management of explicit knowledge.

2.2.3 … and actively supported

The IT support depends on the strategy

The IT support required depends on the knowledge-management strategy too. For the codification model something like a library, containing documents and allowing searches, is needed. The personalization model rather needs a system to allow people to find other people.

The organizational culture should encourage knowledge sharing

Knowledge sharing should be encouraged: if codification is the chosen strategy a system such as an electronic repository should be available and incentives should be provided for people to write down what they know, while within a personalization strategy people should be rewarded for directly sharing their knowledge with others.

Hansen, Nohria and Tierney [22] also point out that knowledge management should not be isolated: coordination between HR, IT and competitive strategy management is essential. This coordination requires the leadership of the general manager (i.e. CEO).

In fact, actively choosing a knowledge management strategy can led to benefits for both the company and its customers, but without a strong leadership no strategy can be chosen nor implemented, either any resistance can be overcome.

The environment: barriers to overcome

As already mentioned before, no approach to knowledge transfer would ever work without a proper organizational environment. Barriers have to be overcome and enablers for the transfer must be provided.

Technology as an enabler is pretty easy to provide: many software solutions are available. Despite its helpful role, technology is nevertheless the ultimately solution: as O’Dell and Grayson [36] point out, the whole information about a process is too complex to be electronically collected. Moreover, access to information is not the main barrier to change.

They list some of the lesson learned about what should be the realistic expectations about networking:

- The really important and useful information for improvement is too complex to put on-line. Technology should not aim to replace the existing sharing process or try to provide the “right answer”, but should instead enhance and support the process.

- There has to be a framework for classifying information, so that different business units can talk to each other using a “common language”.

- Entering information into the system must be part of someone’s job.

- Culture and behaviours are the key drivers and inhibitors of internal sharing. Technical issues aren’t the problem, people and how they use the system are.

Rewarding and support

An important aspect in order to promote a culture of sharing is rewarding. The main point is that rewarding should not be something artificial: the practice of sharing will spread only if it is helpful for people to do their work better. That’s why formal financial rewards are not so used to motivate sharing behaviours: recognizing and celebrating behaviours characterized by expertise sharing should be the way.

The managers/leaders should be convinced of the importance of sharing, so that they can have a supportive role. Examples of support tactics are suggested by Ken Derr of Chevron [36]: tell success stories, remove barriers such as NIH syndrome, reward positive behaviours, show commitment.

2.3 Methods and techniques to manage knowledge

2.3.1 Management: a set of activities

Management means carrying out a set of activities

As the name hints, knowledge management is the management of the knowledge as a resource.

As Wielinga et al. [65] state, the knowledge management activities can be seen as “knowledge-intensive problem solving tasks”. Knowledge, indeed, is not only the object managed, but also a tool required: some knowledge is needed to describe, develop and maintain knowledge.

Wiig et al. [66] but also Schreiber et al. [47] depict two main aspects of knowledge management: knowledge management level and knowledge object level.

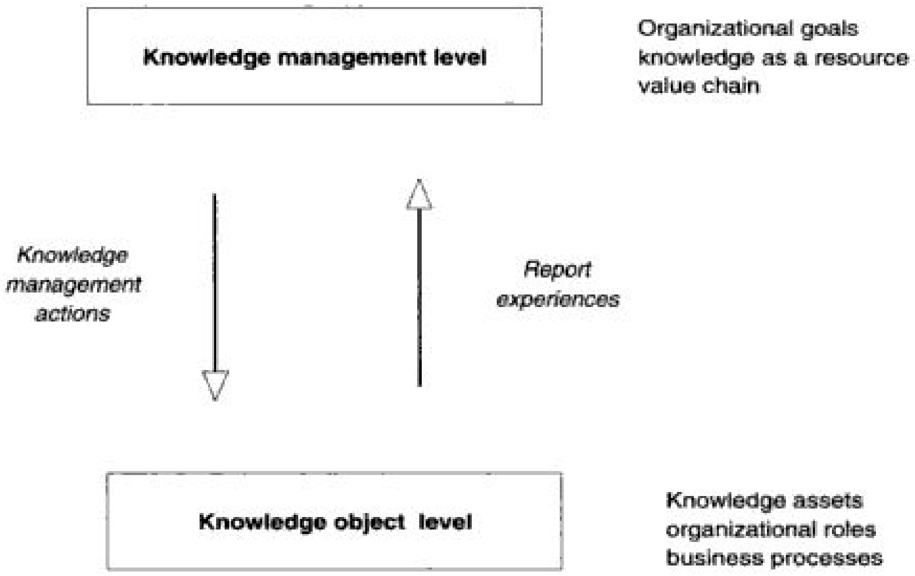


Fig. 2.1 Knowledge management level and Knowledge object level [47]

The knowledge management level

By management (level) we mean that a set of activities take place in order to achieve some kind of goals. These goals consist of both general goals of managing a resource, as knowledge is, and particular goals strictly related to the distinctive characteristics of this particular one.

General goals involve taking care of knowledge for it to be delivered on time at the right place, in the right shape, having the proper quality and being obtained at the lowest possible costs.

Distinctive characteristics of knowledge to which managers should take care are listed: knowledge is

- intangible, difficult to measure
- volatile and strictly dependent from the vessel (i.e. agent with will in whom knowledge is embodied)
- not consumed in a process, may increase through use, can be used by different processes at the same time
- cannot be bought on the market and has long lead times
- has a wide impact on the organization.

The knowledge object level

The object level, on the other hand, is made up by three main components: agents, business processes and knowledge assets. Knowledge management actions operate on these.

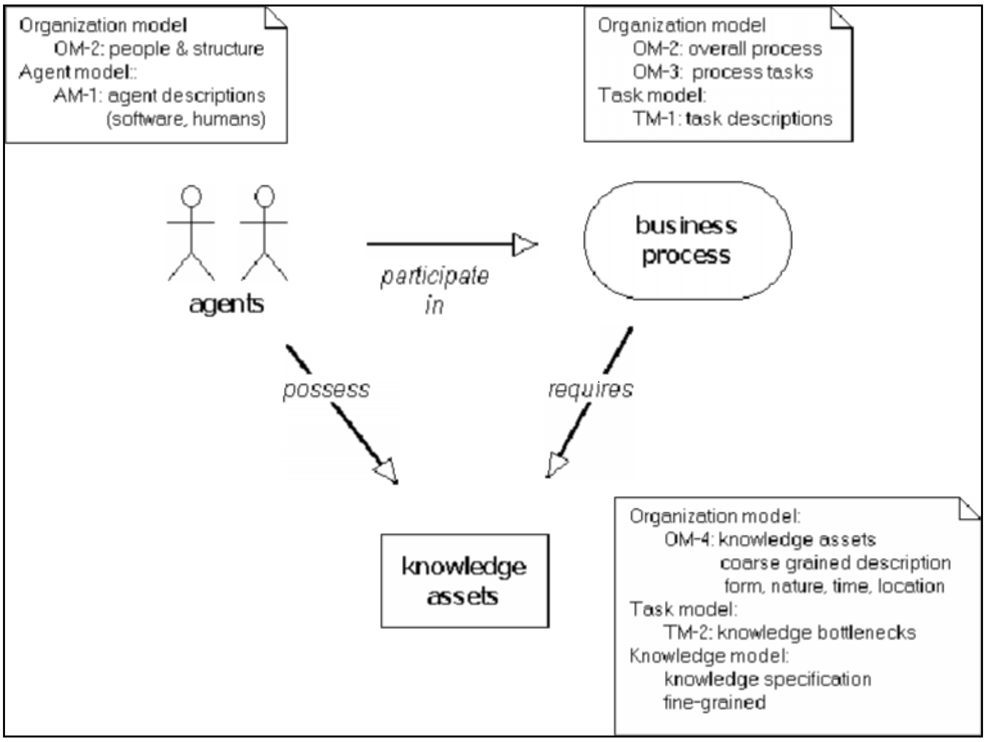


Fig. 2.2 Components of the object level [47]

2.3.2 The knowledge-management cycle

The management level mentioned before takes place within the so-called knowledge management cycle. It is made up of four activities: review, conceptualize, reflect, act. These activities may involve one or a combination of generic operations: developing, distributing, combining and consolidating knowledge.

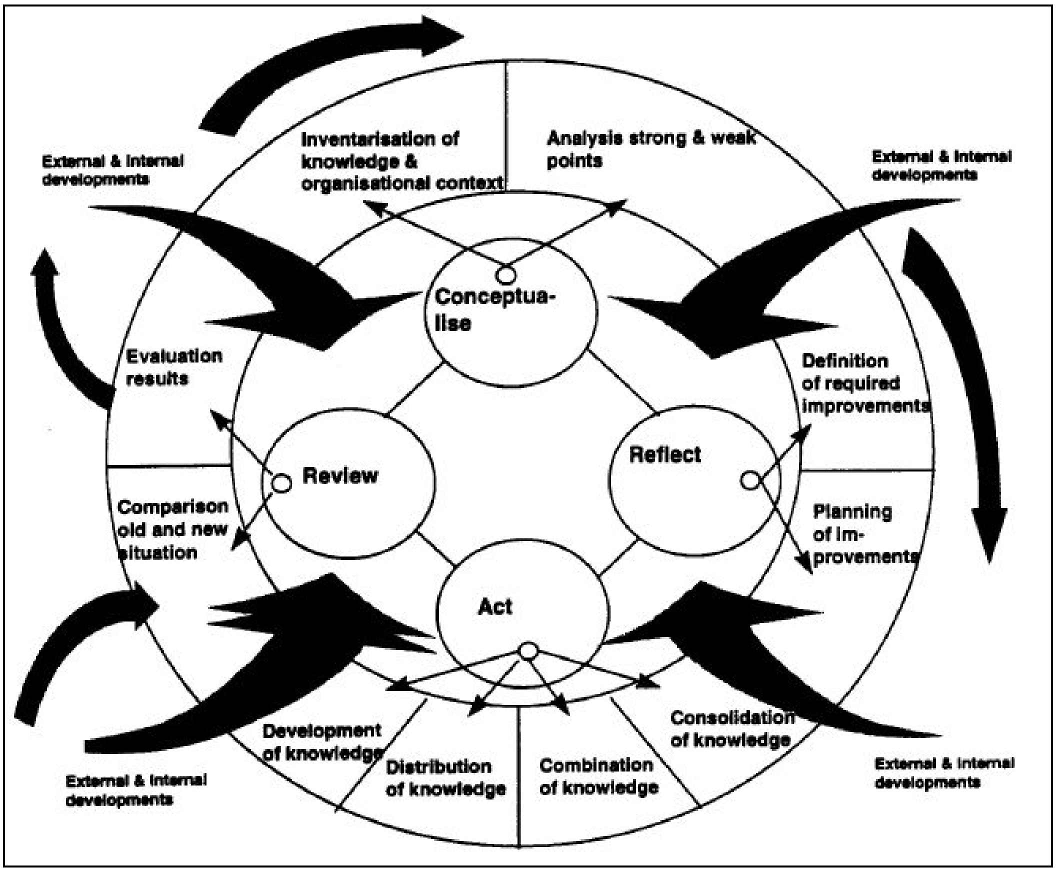

External & internal developments
Inventarisation of knowledge & organisational context
Analysis strong & weak points
External & internal developments
Conceptua-lise
Evaluation results
Definition of required improvements
Review
Reflect
Comparison old and new situation
Planning of im-provements
Act
Development of knowledge
Distribution of knowledge
Combination of knowledge
Consolidation of knowledge
External & internal developments
External & internal developments

Fig. 2.3 The knowledge-management cycle [66]

The knowledge manager (i.e. everyone that happens to manage knowledge) can draw from methods and techniques depicted hereinafter within the description of the phases of the knowledge management cycle.

Review

This activity involves checking out the current situation. It consists of two sub-activities: monitoring and evaluating performance.

Monitoring performance includes all those procedures for monitoring improvement plans and external environment. A SWOT analysis will help.

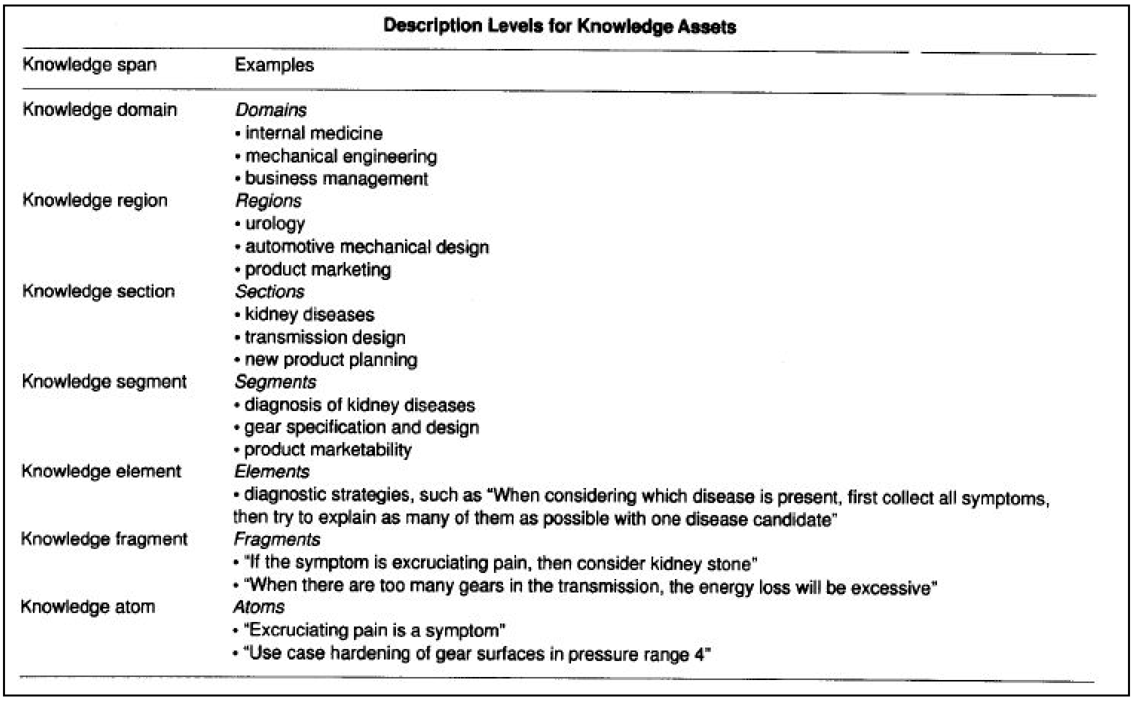
Evaluate performance means that a comparison between original objectives and current situation should be done. The question that should be answered is about where the organization is going from a strategic perspective, both considering the fundamental organizational strategies and the knowledge management strategies.

This first phase of review, described in Wiig et al. [66]'s article, is missing in the description of the knowledge management cycle given from Schreiber et al. [47], but it's reported here because it may be helpful: a better understanding of the current situation should facilitate the undertaking of the following steps.

Conceptualize

This activity aims to answer to which processes use knowledge, which knowledge assets are involved, where and when knowledge is used and which organizational roles provide it.

To answer to these questions, a knowledge inventory has to be done. A convenient way to identify, collect and organize knowledge assets is suggested by Wiig [66]: he proposes a table, "Description Levels for Knowledge Assets", where knowledge levels are ordered from general to specific. The knowledge inventory should focus on the levels in the middle, knowledge section and knowledge segment.

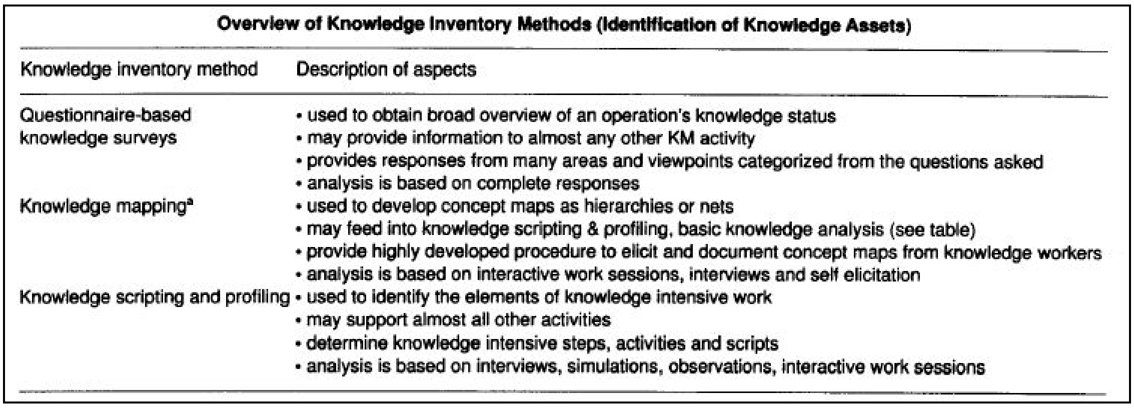


Description Levels for Knowledge Assets

Knowledge span	Examples
Knowledge domain	*Domains* • internal medicine • mechanical engineering • business management
Knowledge region	*Regions* • urology • automotive mechanical design • product marketing
Knowledge section	*Sections* • kidney diseases • transmission design • new product planning
Knowledge segment	*Segments* • diagnosis of kidney diseases • gear specification and design • product marketability
Knowledge element	*Elements* • diagnostic strategies, such as "When considering which disease is present, first collect all symptoms, then try to explain as many of them as possible with one disease candidate"
Knowledge fragment	*Fragments* • "If the symptom is excruciating pain, then consider kidney stone" • "When there are too many gears in the transmission, the energy loss will be excessive"
Knowledge atom	*Atoms* • "Excruciating pain is a symptom" • "Use case hardening of gear surfaces in pressure range 4"

Tab. 2.1 Description levels for knowledge assets [66]

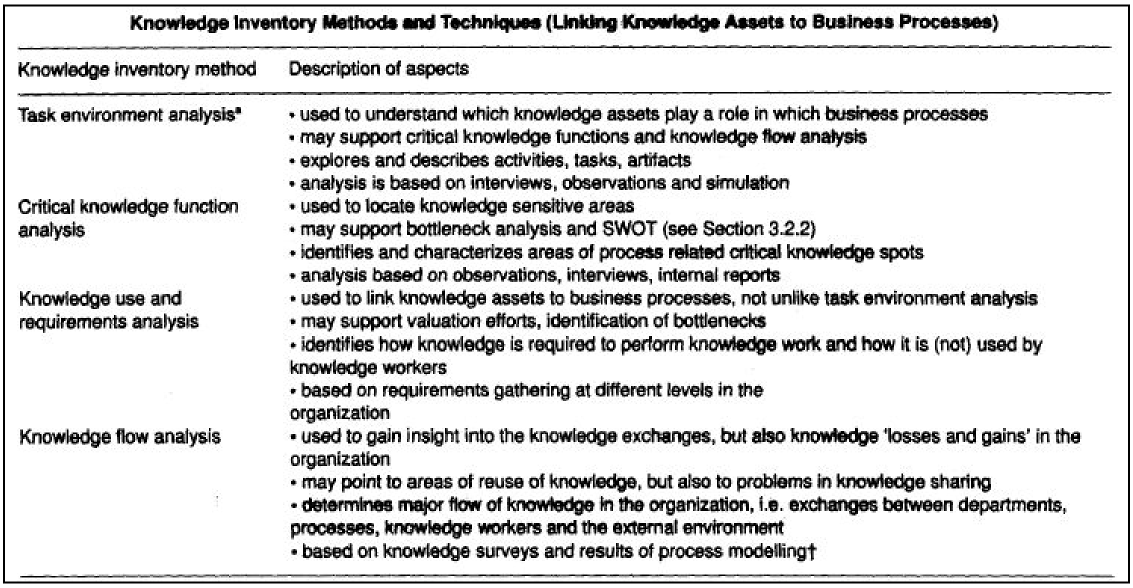
Wiig [66] also proposes more tables as methods to identify the knowledge assets and to link them to business processes.



Overview of Knowledge Inventory Methods (Identification of Knowledge Assets)

Knowledge inventory method	Description of aspects
Questionnaire-based knowledge surveys	• used to obtain broad overview of an operation's knowledge status • may provide information to almost any other KM activity • provides responses from many areas and viewpoints categorized from the questions asked • analysis is based on complete responses
Knowledge mapping[a]	• used to develop concept maps as hierarchies or nets • may feed into knowledge scripting & profiling, basic knowledge analysis (see table) • provide highly developed procedure to elicit and document concept maps from knowledge workers • analysis is based on interactive work sessions, interviews and self elicitation
Knowledge scripting and profiling	• used to identify the elements of knowledge intensive work • may support almost all other activities • determine knowledge intensive steps, activities and scripts • analysis is based on interviews, simulations, observations, interactive work sessions

Tab. 2.2 Methods to identify knowledge assets [66]

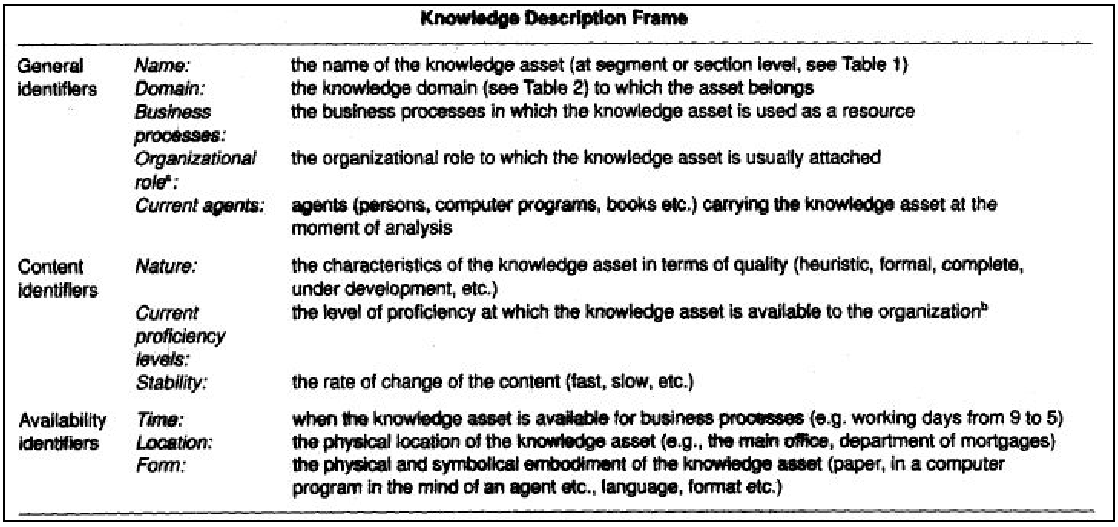


Knowledge Inventory Methods and Techniques (Linking Knowledge Assets to Business Processes)

Knowledge inventory method	Description of aspects
Task environment analysis[a]	• used to understand which knowledge assets play a role in which business processes • may support critical knowledge functions and knowledge flow analysis • explores and describes activities, tasks, artifacts • analysis is based on interviews, observations and simulation
Critical knowledge function analysis	• used to locate knowledge sensitive areas • may support bottleneck analysis and SWOT (see Section 3.2.2) • identifies and characterizes areas of process related critical knowledge spots • analysis based on observations, interviews, internal reports
Knowledge use and requirements analysis	• used to link knowledge assets to business processes, not unlike task environment analysis • may support valuation efforts, identification of bottlenecks • identifies how knowledge is required to perform knowledge work and how it is (not) used by knowledge workers • based on requirements gathering at different levels in the organization
Knowledge flow analysis	• used to gain insight into the knowledge exchanges, but also knowledge 'losses and gains' in the organization • may point to areas of reuse of knowledge, but also to problems in knowledge sharing • determines major flow of knowledge in the organization, i.e. exchanges between departments, processes, knowledge workers and the external environment • based on knowledge surveys and results of process modelling†

Tab. 2.3 Methods to link knowledge assets to business processes [66]

A knowledge description frame can then be implemented: a knowledge object level can be filled.



Knowledge Description Frame		
General identifiers	*Name:*	the name of the knowledge asset (at segment or section level, see Table 1)
	Domain:	the knowledge domain (see Table 2) to which the asset belongs
	Business processes:	the business processes in which the knowledge asset is used as a resource
	Organizational role[a]*:*	the organizational role to which the knowledge asset is usually attached
	Current agents:	agents (persons, computer programs, books etc.) carrying the knowledge asset at the moment of analysis
Content identifiers	*Nature:*	the characteristics of the knowledge asset in terms of quality (heuristic, formal, complete, under development, etc.)
	Current proficiency levels:	the level of proficiency at which the knowledge asset is available to the organization[b]
	Stability:	the rate of change of the content (fast, slow, etc.)
Availability identifiers	*Time:*	when the knowledge asset is available for business processes (e.g. working days from 9 to 5)
	Location:	the physical location of the knowledge asset (e.g., the main office, department of mortgages)
	Form:	the physical and symbolical embodiment of the knowledge asset (paper, in a computer program in the mind of an agent etc., language, format etc.)

Tab. 2.4 Guidelines for a knowledge description frame [66]

Within the conceptualization phase, an analysis of strong and weak point has to be done too. Methods such as bottleneck analysis and SWOT analysis can be used.

Some guidelines for this phase are suggested by Schreiber et al. [48]:

- Find a proper scope for the conceptualization. Good starting points are initial bottlenecks, new business opportunities, human resource problems.
- Choose the proper level of detail (the middle levels of Wiig’s table, as hinted before).
- Be aware of hidden knowledge, especially “informal” knowledge that everybody takes for granted.
- Never rely on a single source when trying to link knowledge to agents. A network analysis should be done.
- Beauty is in the eye of the beholder: different viewpoints should be alternated.
- Some quantification is better than no quantification at all.

Wielinga et al. [65] propose another way of dealing with knowledge inventories: using libraries of ontologies. Ontologies can play a role in knowledge management at different levels:

- Object level: support the accessibility of knowledge through representations and indexing of information
- Domain level: modelling of business process to support the management

- Content level: as an agreement over terminology.

Reflect

The main goal of this activity is to produce improvement plans to be executed within the Act phase. So the aim of this phase is to point out what improvements should be made and how to actually make them.

It’s important to define and select the correct improvements: solving the wrong problem and selecting the wrong solution won’t help. This task is anything but easy. Wiig et al. [66] suggest that a good approach may be thinking in terms of programs rather than in terms of action. Programs should concern effectiveness improvement, knowledge building and strategic action. The SWOT analysis can be used as a guide. Once defined, the improvements should be prioritized. The suggested approach is MAUT, Multi-Attribute Utility Theory: this method evaluates the selected improvements through a set of attributes that are important for the decision maker.

Later on, operational plans based on the chosen improvements should be made. Schedules, budget, deliverables, people involved and quality affect the planning. Also, responsibilities should be considered and risks should be assessed.

In this phase knowledge management starts to diverge from knowledge engineering, as it emerges from Schreiber et al. [48]’s guidelines:

- Take a maximum distance from methodologies such as CommonKADS, to prevent solutions from being dependent on them.

- Avoid the trapdoors of “solving the wrong problem” and “selecting the wrong solution” (as explained before).

- There are no silver bullets: organizations and knowledge are too complex for just one single measure to be the one.

- Abide by Murphy’s law: be very aware of risks.

- Sleep on it: review the process.

Act

This activity concerns the running of the improvement plans. It’s actually beyond the scope of knowledge management: it belongs more to the jurisdiction of HR, IT and Organization Development.

By the way, even for this phase two main guidelines are suggested from Schreiber et al. [48]:

- Go for measurable objectives.
- Things do not run themselves: assign clear responsibilities, give clear briefs and carry out frequent control on progress.

2.3.3 Creating value: the knowledge value chain

A knowledge value chain can be detected looking at the activities in knowledge management:

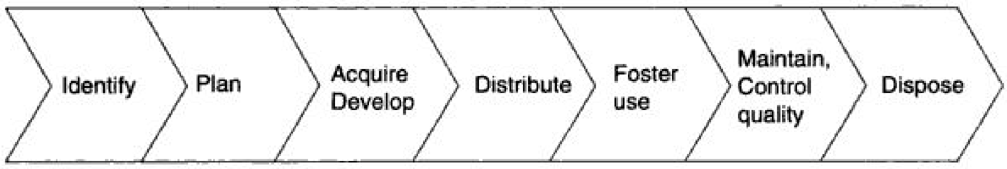


Fig. 2.4 Knowledge value chain [46]

The internal and external existing knowledge is identified. It is planned what knowledge will be needed in the future: this knowledge is acquired, developed and distributed were needed. The use of knowledge is fostered, its quality is controlled and maintained. Knowledge is disposed when no longer needed.

The activities shown in the picture form a coherent whole which, by analogy with the famous Porter's value chain, is called knowledge value chain.

Therefore, as long as the value chain is identified, knowledge management can be defined as "a framework and tool set for improving the organization's knowledge infrastructure, aimed at getting the right knowledge to the right people in the right form at the right time" [46].

To create value for the final customer business processes have to take place: knowledge is the enabler for processes to be successfully carried out.

That's how a strategy for knowledge management is formulated: it considers the value-creation goals, it analyzes how these goals are reached by the business processes and in the end elaborates on which knowledge is embodied within these processes. To do this, many managerial actions can be undertaken, a wide variety of methods is available.

Some useful lessons should be retained:

- Knowledge is an organizational asset but it resides in individuals.

- Knowledge is what is actually done, is a potential for action.
- The knowledge management strategy should be directed from outside-in.
- Knowledge sharing relies on communication between individuals and knowledge management should facilitate this through increasing people's connectivity.

2.4 Knowledge engineering to support knowledge management

2.4.1 Knowledge engineering and knowledge systems

Knowledge engineering and its benefits

A wide range of methods and techniques goes by the name of knowledge engineering. These methods aim to be a tool for the acquisition, modelling, representation and use of knowledge [65].

Benefits are related to the whole discipline of knowledge engineering. Knowledge engineering provides methods for better understanding the structure and the processes used by knowledge workers. It spots opportunities and bottlenecks, giving managers some useful tools for a better integration of IT in support of knowledge work and for a more effective corporate knowledge management.

Knowledge systems

Knowledge engineering is the discipline that produces knowledge systems to help human problem-solving.

The benefits of knowledge systems are described within the empirical study of Martin et al. [44], where two questions are addressed:

- What are benefits expected from the use of knowledge systems?
- Are expected benefits from an investment in knowledge systems actually realized?

These questions were answered through the collection of survey data from business people, that is people who directly benefit from the use of knowledge systems.

The top three benefits detected are:

- faster decision making
- increased productivity
- increased quality of decision making.

These benefits, as the authors of the survey point out, occur in varying degrees. What is worth to notice is that knowledge systems actually improve the organizational effectiveness.

A structured approach to software development: the pyramid

A structured approach to analysis, design and management is necessary for knowledge systems.

Every software development approach consists of a number of elements, commonly represented as a pyramid. The building blocks, starting from the bottom of the pyramid, are: worldview, theory, methods, tools, use.

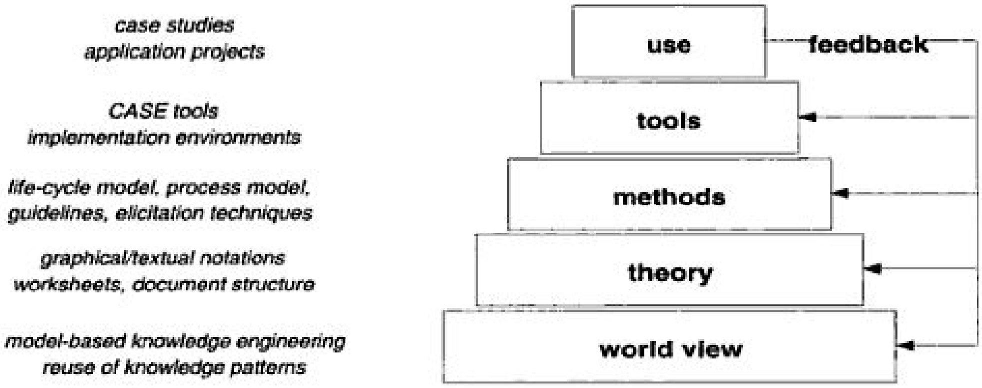

case studies
application projects
CASE tools
implementation environments
life-cycle model, process model,
guidelines, elicitation techniques
graphical/textual notations
worksheets, document structure
model-based knowledge engineering
reuse of knowledge patterns
use
feedback
tools
methods
theory
world view

Fig. 2.5 A structured approach to software development [45]

The lowest step, worldview, is a set of principles that constitute the slogans of each approach. These slogans are then translated into theoretical concepts, methods for using the methodology and tools for applying them and, in the end, case studies that collect experiences from the use of the methodology, as the pyramid depicts. Feedback winds around each step.

Nature and locus of knowledge: attributes to define knowledge-management systems

Hahn and Subramani [20] have conducted a series of semi-structured interviews with knowledge managers in order to provide a framework of the different approaches to knowledge-management systems. The two attributes that define the kind of system in use are the nature and the locus of knowledge.

The locus of the knowledge is where the knowledge resides: it can be an artefact (e.g. a document) or an individual. The nature of the knowledge describes the level of a priori structure imposed by the knowledge-management system (i.e. structured or unstructured).

The following picture shows some example of knowledge-management systems with different locus and nature of knowledge.

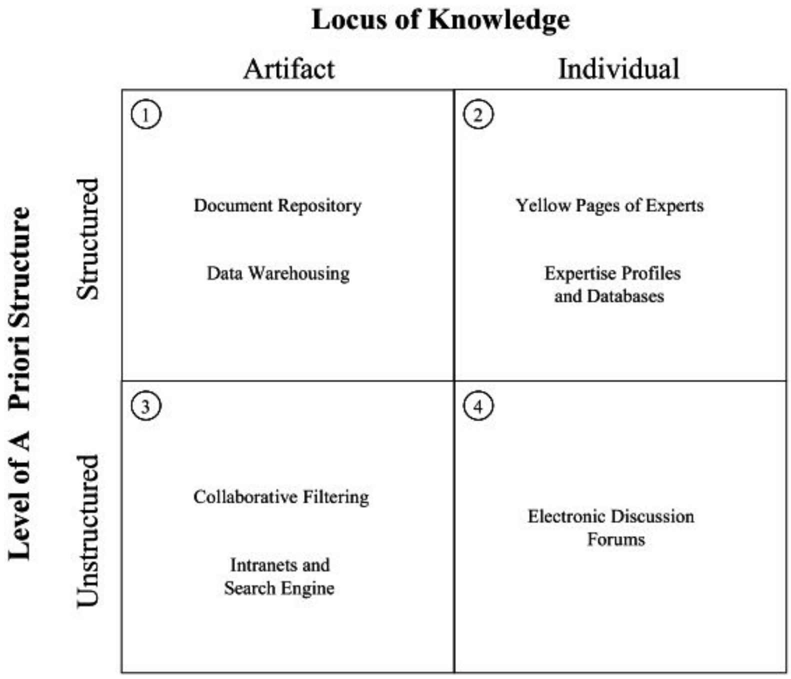


Fig. 2.6 Examples of knowledge-management systems according to structure and locus of knowledge [20]

Cell 1 is about those systems for managing the organizational knowledge that is or can be codified. Cell 2 concerns knowledge that resides in individuals, but managed through categorizing schemes. Cell 3 comprises systems where knowledge is captured in artifacts, but there is no categorization. Cell 4 is about those systems which allow the users to look for others that can help them.

Knowledge-management systems: balance, maintenance, development

Some important considerations about knowledge-management systems:

- Size can have a positive or negative network effect. If the knowledge sources are artefacts, the greater their number is, the higher the chances to find a document of interest are. If the knowledge sources are individuals increasing size may lead to overload.

- The diversity of content is not a problem when the knowledge sources are highly structured, but it could be when they are not (e.g. leading to an incongruent vocabulary)

Therefore, the common theme underlying these systems is to provide a technical solution to avoid information overload and yet support and facilitate the location of useful content through allowing a proper growth of the system.

A critical problem is motivating users to contribute: motivation is influenced by the efforts and time required to use the system, especially for what concerns the structuring of the contribution. Linked to this there is the risk that all the work ends up as a responsibility of a few group of experts, which in turn will be overloaded because of the increased burden. Solutions may be taking turns in using the system or, at the opposite, creating a new role such as the knowledge librarian within the organization.

Hahn and Subramani [20] take a look into the long-term effects of knowledge-management systems. They highlight how the use of knowledge-management systems can lead to both positive and negative outcomes, as the organization may gain in efficiency through reuse of knowledge but may also become rigid and loose the capacity to learn and innovate as long as it relies on existing solutions.

Hahn and Subramani [20] also elaborate on the difficulty of developing these systems: it’s hard to know a priori what information will be requested, who will be the user and who will be the supplier and when and how the information will be used. So it’s difficult to define a typical user and the system has to be flexible: that’s why an evolutionary approach to system development should be followed.

The most important thing to remember is that “a tool is only successful if the users of the tools succeed with the tool” [20]: motivating users and helping them to accept the system becomes critical.

2.4.2 CommonKADS

What is CommonKADS?

CommonKADS, as described in the related website [68], is the leading methodology to support structured knowledge engineering. It is the European de facto standard for knowledge analysis and knowledge-intensive system development, and it has been adopted by many major companies in Europe, as well as in the US and Japan.

The CommonKADS methodology helps knowledge managers in spotting opportunities and bottlenecks within the management of knowledge in organizations, that is how organizations develop, distribute and apply their knowledge resources, and enables to perform a detailed analysis of knowledge-intensive tasks and processes. Through this methodology, knowledge systems development is supported; in particular, CommonKADS suits the object-oriented development and uses notations compatible with UML. Therefore CommonKADS is useful both for knowledge managers and software engineers.

People interested in knowledge management often find that there is a lack of support techniques for practical day-to-day knowledge management: CommonKADS is a powerful tool which helps the knowledge manager in defining a corporate knowledge-management strategy through knowledge analysis and knowledge system development.

The core of CommonKADS is formed by its knowledge analysis framework: knowledge-intensive tasks can be analyzed at different grain-size levels thanks to the support of "templates", predefined reusable knowledge models of proven soundness. The results of knowledge analysis are documented in the "knowledge model", a specification of the information and knowledge structures involved in a knowledge-intensive task.

Results of knowledge analysis can be used as specifications for the development of a knowledge system. CommonKADS is useful especially within the early stages of system development, since it provides a clear route to implementation. It also provides tools and techniques for feedback, prototyping, etc.

The CommonKADS methodology: knowledge engineering or knowledge management?

Based on Newell’s concept of knowledge level, the level of knowledge, implementation-independent, that provides a conceptual description of problem solving behaviour and of those knowledge structures that sustain that behaviour, a number of knowledge modelling methodologies have been developed: one of these is CommonKADS [65].

Describing a process model for the management level and an object model for the object level so that knowledge management becomes effective is something very similar to what CommonKADS does for knowledge engineering. Furthermore, the components affected by the knowledge management actions are very similar to what is called context of a knowledge system in CommonKADS models. That’s why this methodology is analyzed in the present work.

Even if there is a seamless link between knowledge management and knowledge engineering, they are different because they belong to different organizational roles. Knowledge systems should be considered as tools for knowledge management, but building them is something that should be delegated to knowledge engineering. However, the CommonKADS methodology numbers among its merits the possibility of being shared between the knowledge manager (i.e. the person in charge for knowledge management) and the project manager (in charge of knowledge engineering).

Main principles of the CommonKADS methodology

The CommonKADS methodology relies on several principles, based on the lessons learned about knowledge system development in the past. Schreiber et al. [45] list the fundamentals:

- Knowledge engineering is no longer a process of extracting knowledge from an expert's head to transfer it: today it's approached as a modelling activity. Modelling means that a focus on few aspects of knowledge, the useful ones, is carried out. The CommonKADS model is a good tool to structure this activity.

- According to Newell's knowledge level principle [45], knowledge should be modelled at a conceptual, implementation independent, level. Programming details should be left for later. In the CommonKADS view this is called structure-preserving design.

- Knowledge has a stable internal structure and it can be analyzed through the identification of knowledge types and roles: knowledge is depicted as a well-structured functional whole with different roles for each part in human problem solving. The main roles are:

    - knowledge provider/specialist, owner of knowledge, expert in the application domain

    - knowledge engineer/analyst, in charge of system-analysis work

    - knowledge-system developer, responsible for design and implementation

    - knowledge user, who directly or indirectly uses the knowledge system

    - project manager, in charge of running the knowledge system development project (he is likely to benefit from a structured approach as CommonKADS)

    - knowledge manager, not directly involved in the projects, formulates a knowledge strategy at the business level.

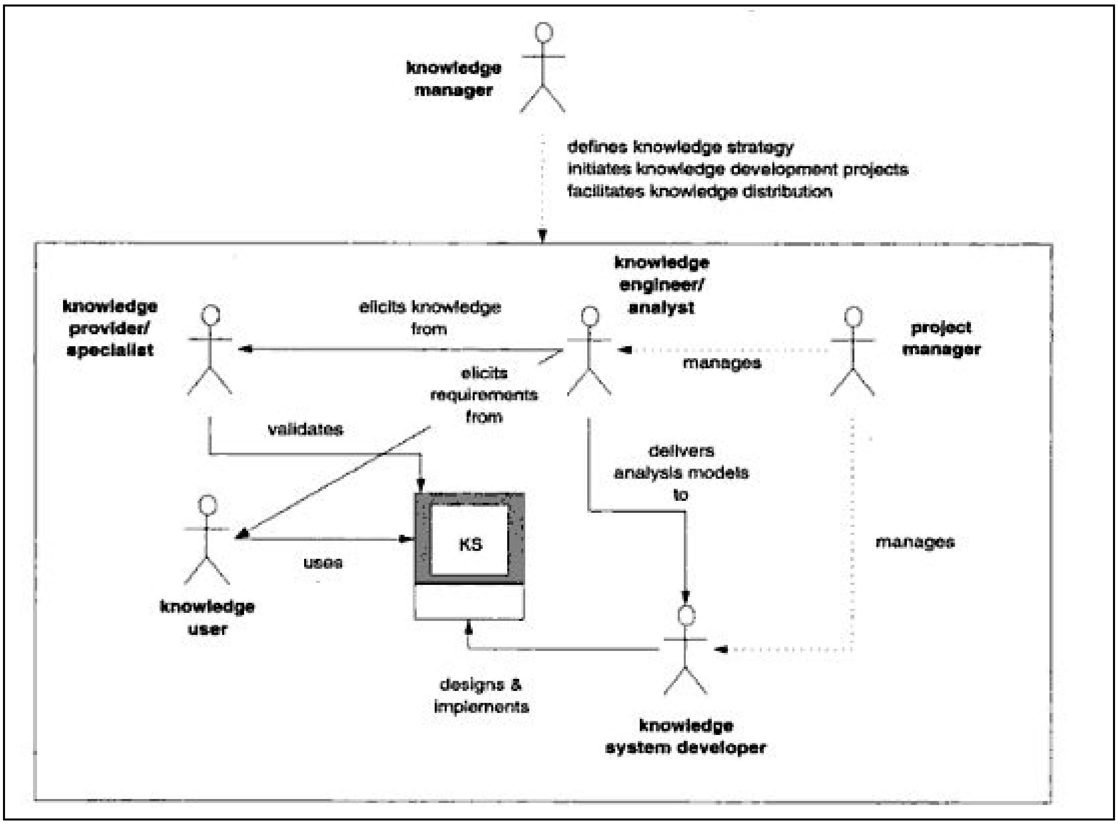

knowledge manager
defines knowledge strategy
initiates knowledge development projects
facilitates knowledge distribution
knowledge engineer/ analyst
knowledge provider/ specialist
elicits knowledge from
elicits requirements from
project manager
manages
validates
delivers analysis models to
knowledge user
uses
KS
manages
designs & implements
knowledge system developer

Fig. 2.7 Knowledge roles according to the CommonKADS methodology [45]

- A knowledge project must be managed learning from experience in a controlled “spiral” way. A waterfall approach is too rigid, while prototyping, even if very popular, lacks of control. CommonKADS offers a structured way of learning, more flexible than the waterfall model and more controlled than rapid prototyping.

The CommonKADS model suite

Schreiber et al. [45] present an overview of the CommonKADS model suite, which is the core of the CommonKADS knowledge-engineering methodology.

The suite spreads over three levels:

- Context – In this level should be explained why the knowledge system is chosen and which impacts it has on the organization.

- Concept – Here the nature and structure of the knowledge involved and of the corresponding communication are analyzed.

- Artefact – The main focus here is on how the knowledge is implemented in the computer system and how the software architecture looks like.

CommonKADS provides a set of models that help developing an overall view of the organizational environment, to report the critical success factors for the knowledge system, to describe the problem-solving functions and data that belong to the system itself and to deliver the technical specifications for its implementation.

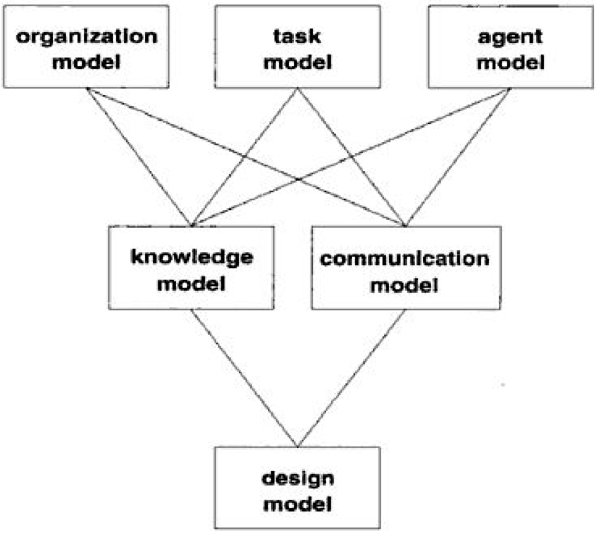


Fig. 2.8 CommonKADS models for an overall view of the organizational environment [45]

As shown in the picture, models are:

- the organizational model – for the analysis of the major features of the organization
- the tasks model – to identify the relevant subparts of each business process
- the agent model – to describe the executors of the tasks
- the knowledge model – to explicate in detail the types and structures of the knowledge used in performing a task
- the communication model – to highlight the communicative transactions between the agents involved
- the design model – to give the technical systems specifications.

The deliverables produced by a CommonKADS knowledge project are model documents, project management information and a knowledge system software.

3 Managing a Software Company

3.1 Managing processes

3.1.1 Software processes

Defining a software process and its activities

A software process is a set of activities that leads to the production of a software product. It relies on people making decisions and judgements and that is why there is no ideal process and many organisations have developed their own approach to software development.

Even if there are many software processes, some fundamental activities are common to all software processes: software specification, software design and implementation, software validation, software evolution [49]. These activities are differently organized depending on the different development process.

Software specification (also known as requirements engineering) is the process of understanding and defining what the system should do and what are the constraints to its development. A requirements document, that is the specification for the system, is produced; it is made of four intertwined phases: feasibility study, requirements elicitation and analysis, requirements specification and requirements validation.

Software design and implementation is the process of converting a system specification into an executable system; it involves processes of software design and programming, but may also involve refinement of the software specification. Design activities are: architectural design, abstract specification, interface design, component design, data structure design, algorithm design. The output of each activity is a specification for the next stage.

Software validation (also called V&V, verification and validation) is the process of checking whether a system matches its specification and the customer's expectations. It involves inspections and reviews at each stage of the software process, even if testing is the most expensive stage. The testing process should be iterative and continuous feedback to the earlier stages should be provided; testing stages concern the components (units), the system and the acceptance (alpha testing).

Software evolution (or maintenance) is about the flexibility of software systems: changes can be made to software at any time, but the cost of maintenance is often several times the initial development cost. However, the distinction between development and maintenance is vanishing in favour of the so-called software engineering ad evolutionary processes.

Models of software processes

A software process model is an abstract representation of a software process that explains the chosen approach to software development.

Models are [49]:

- The waterfall model (or software life cycle), where fundamental activities (requirement analysis and definition, system and software design, implementation and unit testing, integration and system testing, operation and maintenance) are represented as separate process phases. Each phase leads to an approved document and allows the beginning of the next phase, even if in practice the stages may overlap and feed information to each other. Iterations are costly and involve significant rework and that's why parts of the development often are frozen. This model should be used when the requirements are well understood and unlikely to change radically during system development.

- Evolutionary development, where the fundamental activities are interleaved and an initial system is rapidly developed and then refined. There are two types of evolutionary development: exploratory development, where the system evolves following the customer hints, or throwaway prototyping, where the objective is to develop a better requirements definition. The evolutionary model is more effective than the waterfall model since the specification can be developed incrementally, but the process is not visible and systems are often poorly structured.

- Component-based software engineering, based on the existence of reusable components and some integrating framework for these components. After the initial requirements specification stage the following stages take place: component analysis, requirements modification, system design with reuse, development and integration and finally system validation. This model reduces the amount of software to be developed and so reduces costs and risks and may lead to a faster delivery. However, requirements compromises are inevitable.

In practice, these models are often combined.

Process iteration

Since change is inevitable in the software process, two models have been explicitly designed to support process iteration [49]:

- Incremental delivery.

  Advantages: customers can gain value from the system before the whole system is delivered, they can use the early increments as prototypes, there is a lower risk of overall project failure and the most important system services, developed first, receive the most testing.

  Problems: increments should be relatively small and each one should deliver some system functionality, not well defined requirements lead to a difficulty in identifying common facilities.

- Spiral development.

  The phases are: objective setting, risk assessment and reduction, development and validation, planning. The recognition of risk is explicit.

  The essence of iterative processes is that the specification is developed in conjunction with the software: there is no complete system specification until the final increment is specified.

3.1.2 ISO/IEC 15504 aka SPICE

History

ISO/IEC 15504 [76], also known as SPICE (Software Process Improvement and Capability Determination), is a framework for the assessment of processes developed by the conjoint effort of ISO, International Organization for Standardization, and IEC, International Electrotechnical Commission.

It is an international standard derived from process lifecycle standard ISO 12207 and maturity models such as CMM and aims to be a reference model for the assessment of organizational capabilities for delivering products (software, systems, IT services).

The acronym SPICE initially stood for "Software Process Improvement and Capability Evaluation", but because of French concerns over the meaning of "evaluation", it has been redefined as "Software Process Improvement and Capability Determination".

The first versions of the SPICE standard focused exclusively on software development processes, but later it was expanded to cover all the processes related to a software business. In particular, six business areas are covered: organizational, management, engineering, acquisition supply, support, operations.

Reference model

ISO/IEC 15504 contains a reference model in which a process dimension and a capability dimension are defined.

The process dimension describes each process as it belongs to one of the five process categories:

- customer-supplier
- engineering
- supporting
- management
- organization.

The capability level of a process is rated on the following scale:

5 Optimizing process

4 Predictable process

3 Established process

2 Managed process

1 Performed process

0 Incomplete process.

The capability of a process is measured using process attributes; the international standard defines nine process attributes:

1.1 Process Performance

2.1 Performance Management

2.2 Work Product Management

3.1 Process Definition

3.2 Process Deployment

4.1 Process Measurement

4.2 Process Control

5.1 Process Innovation

5.2 Process Optimization.

Each process attribute consists of one or more generic practices, which are further elaborated into practice indicators to aid assessment performance. Each process attribute is assessed on a four-point (N-P-L-F) rating scale:

Not achieved (0 - 15%)

Partially achieved (>15% - 50%)

Largely achieved (>50%- 85%)

Fully achieved (>85% - 100%).

The rating is based upon evidence collected against the practice indicators, which demonstrate fulfillment of the process attribute.

Assessment process

ISO/IEC 15504 provides a guide for performing an assessment, including the assessment process, the model for the assessment and the tools used during the assessment. The general steps of the assessment are the following:

1. Initiate an assessment (assessment sponsor).
2. Select assessor and assessment team.
3. Plan the assessment, including processes and organizational unit to be assessed (lead assessor and assessment team).
4. Pre-assessment briefing.
5. Data collection (through interviews with people who perform the process and through collecting documents, quality records and statistical process data).
6. Data validation.
7. Process rating (through the assessor judgment, against process's base practices and the capability dimension's generic practices).
8. Reporting the assessment result to the sponsor.

For an actual assessment, the process assessment model (PAM) is used. It is an elaboration of the process reference model provided by the process lifecycle standards.

Tools for the assessment can be paper-based manually used, which incorporate the assessment model indicators and the base and generic practice indicators. A limited number of computer-based tools is also available.

For the assessment to succeed, the assessor must be qualified and competent. He need to possess a suitable level of the relevant skills and experience such as

- personal qualities (e.g. communication skills)
- relevant education, training and experience
- specific skills for particular categories (e.g. management skills for the management category)
- ISO/IEC 15504 related training and experience in process capability assessments, which comprises a 5 day training course, at least one assessment successfully performed under supervision and one as a lead assessor.

Uses and success of ISO/IEC 15504

ISO/IEC 15504 can be used for both process improvement and supplier's process capability determination.

For what concerns process improvement, ISO/IEC 15504 helps to better understand the initial baseline level (process capability level) and to assess the situation after an improvement. It provides guidance on defining objectives, planning and executing improvements through an eight-step improvement programme.

ISO/IEC 15504 is also used to evaluate the capability of potential suppliers to deliver, which is important for organization considering outsourcing. It provides a framework for assessing proposed suppliers (that can be assessed by the organization itself or by an independent assessor) against a target capability which is based on the organization's needs. This target capability is described by a set of target process profiles, particularly important in contexts where the organization is required to accept the cheapest qualifying vendor. Target capability is useful for suppliers as well: it enables them to identify gaps between their current capability and the level required by their (wanted) customer.

Even if successful, ISO/IEC 15504 has not been as successful as the CMMI. Reasons are the following: it is not available as free download but must be purchased (CMM and CMMI are available as free downloads from the SEI website), the CMMI is actively sponsored and was created first, the CMMI retains the benefits of the CMM and incorporates many of the ideas of ISO/IEC 15504 too.

3.1.3 Managing individuals: the PSP

The Personal Software Process

The Software Engineering Institute has developed the so-called Personal Software Process (PSP), a process to guide improvement in small organizations and project teams based on the idea that personal process discipline can lead to individual effectiveness, which would likely improve teams and projects' performance.

The PSP course aims to help engineers in defining and measuring their process through ten software development exercises that show how effective their methods are and how their performance can be improved. Self study is also possible, but a PSP course seems to be more effective.

Under development at the SEI since 1989, the PSP has been taught within several universities and companies and corporations such as Siemens and Helwett Packard have participated to the study.

Disciplined processes lead to effective methods

As already mentioned, PSP strategy is to motivate engineers to find (and adopt) effective methods through a disciplined, gradual and defined process. This is important in software development because software groups usually do not provide sound engineering methods, professionals' methods are often private and often software people are not trained for planning and evaluating the methods they use. It's also important that each engineer looks for methods for himself, so that he becomes sure of their effectiveness and decides to consistently use them.

Defining, measuring and tracking their own work leads software engineers to a better understanding of their performance, which enables them to recognize best practices and motivates them to look for further improvement. As the members of a team improve, then, the team itself will improve.

The PSP's maturity framework

Humphrey [28] provides examples of effective practices at an individual level in relation to the capability maturity model (CMM):

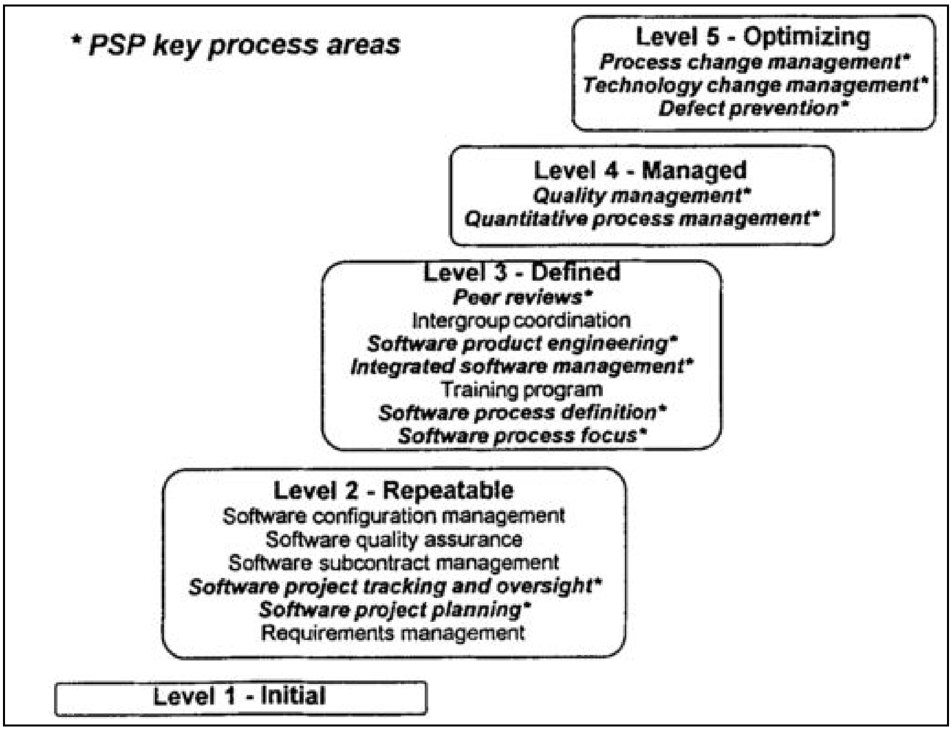

* PSP key process areas
Level 5 - Optimizing
Process change management*
Technology change management*
Defect prevention*
Level 4 - Managed
Quality management*
Quantitative process management*
Level 3 - Defined
Peer reviews*
Intergroup coordination
Software product engineering*
Integrated software management*
Training program
Software process definition*
Software process focus*
Level 2 - Repeatable
Software configuration management
Software quality assurance
Software subcontract management
Software project tracking and oversight*
Software project planning*
Requirements management
Level 1 - Initial

Fig. 3.1 Examples of effective practices in relation to the CMM [28]

The PSP has a maturity framework much like that of the CMM:

- PSP0 (+ PSP0.1): Baseline Personal Process. This initial step is about measurements on the current process used to write software: time, defects, size. Defect and coding standards are also analyzed. The process improvement proposal, i.e. the structured way to record problems, experiences and improvement suggestions, is described too.

- PSP1 (+ PSP1.1): Personal Planning. This step adds size and resource estimation, a test report, schedule planning and status tracking. It aims to teach to engineers about the relationship between programs' size and time to develop them, how to make accessible commitments, how to plan work and how to determine their status.

- PSP2 (+ PSP2.1): Personal Quality Management. The objective of this step is to improve the engineers' ability to produce high-quality programs through using defect data to reduce compile and test defects. Engineers have to learn to deal objectively with their mistakes, understand how many defects they inject and why and with what consequences. Design and code reviews are analyzed.

- PSP3: Cyclic Personal Process. Large programs (i.e. programs of up to several thousand lines of code, KLOC) are usually realized through the integration of smaller PSP2-sized programs: to effectively integrate them, a cyclic development process is followed. So, this step's objective is to introduce the engineers to the principles of process scaling.

As the size of the project increases, PSP3 can be no longer enough and a team project may fit better. PSP, however, does not define team processes.

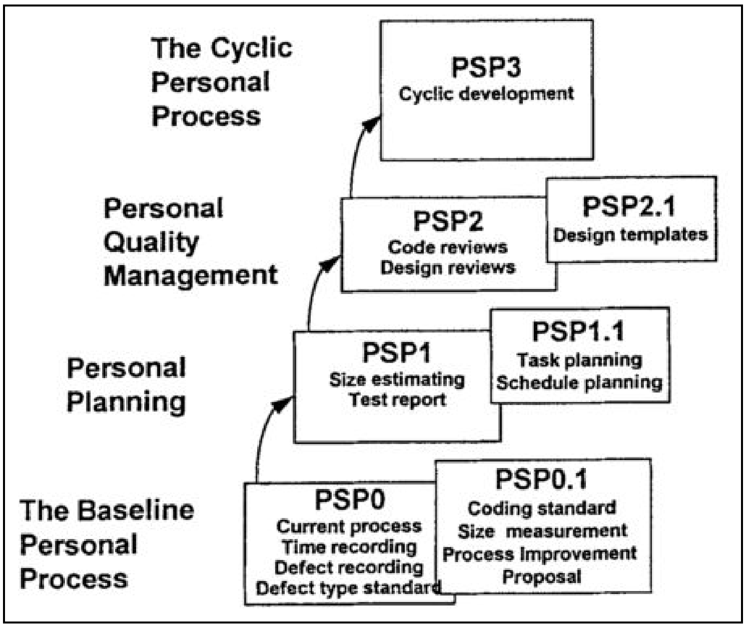


Fig. 3.2 PSP's maturity framework [28]

The PSP's principles

The main PSP principles are the following:

- An efficient work relies upon a defined and structured process. Creative and routine parts of the software process should be treated differently: routine tasks involving defined procedures, forms and historical data should be structured and made more accurate and efficient.

- A defined process must fit the individual skills and preferences of the professional who uses it. The value of using a process is more clear when the process is specific.

- A professional is comfortable with a defined processes only if he is involved in their definition. Being part of the processes' definition is what makes engineers better understanding the effectiveness of the methods used and the importance of improvements and adjustments.

- Processes should evolve together with professional's skills and abilities. One of the main characteristics of the software industry as well as of the software products is the rapid rate of changing and evolution. Software engineers' skills and abilities need to be developed in order to face changes and the processes should evolve hand in hand with them to remain useful.

- A rapid and explicit feedback is the enabler for continuous process improvement. Feedback provides reinforcement to what is being taught. Both long-term and short-term feedback need to be provided.

As Humphrey [28] recaps, PSP as a structured, disciplined and measured software process is a successful tool as it provides engineers with the guidance and feedback required to improve their personal performance.

3.2 Managing people

3.2.1 People: the intellectual capital – a critical issue

Critical factors in managing people

The people working in a software organisation are its greatest assets: that's why this ”intellectual capital” should be respected, given a sense of responsibility and rewarded properly. Poor management of people in an organization is one of the most significant contributors to project failure.

Sommerville [50] identifies four critical factors in the people management:

- Consistency: people shouldn't feel that their contribution to the organisation is undervalued.
- Respect for the differences between different people's skills.
- Inclusion: people should feel listened and that their proposals are taken into account.
- Honesty: managers should tell the truth when answering questions like “What is going well?”, “What is going bad?”, “What is my level of technical knowledge?”.

The choice of people

Often project managers don't have a free choice over the team members, because of budget constrains, because of people's availability (i.e. people may be able to work only for part of the time on a certain project because they have to work on something else too) and also because some skills are in short supply.

The decision on who to appoint to a project usually is made using three types of information [51]:

- about the candidates, their background and experience, from their résumé or CV,
- information gained by interviewing candidates
- recommendations from people who have worked with them.

Motivating people

Motivation means organising the work and the work environment so that people are stimulated to work as effectively as possible. There are many theories about what motivates people.

Maslow [32] suggests that people are motivated by satisfying their needs, which are arranged in a series of level: physiological needs, safety needs, social needs, esteem needs and self-realization needs. People working in software development are not hungry or thirsty and generally do not feel threatened. Therefore, satisfying social, esteem and self-realisation needs is what matters from a management point of view.

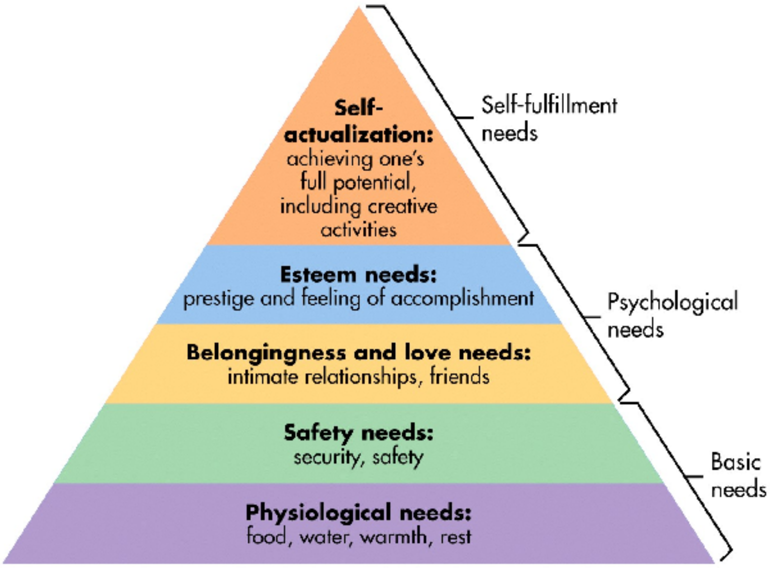

Self-actualization: achieving one's full potential, including creative activities
Esteem needs: prestige and feeling of accomplishment
Belongingness and love needs: intimate relationships, friends
Safety needs: security, safety
Physiological needs: food, water, warmth, rest
Self-fulfillment needs
Psychological needs
Basic needs

Fig. 3.3 Maslow's hierarchy of needs

To satisfy social needs, time and spaces to meet co-workers should be given to people. To satisfy esteem needs, people should be shown to be valued by the organisation. To satisfy self-realisation needs, people should be given responsibility for their work, be assigned demanding but not impossible tasks and be provided with a training programme to develop their skills.

The main problem with Maslow’s model of motivation is that it takes an exclusively personal viewpoint on motivation: it does not properly consider motivational the fact that people feel themselves to be part of an organisation, a professional group and a culture.

Bass and Dunteman [3], instead, classify professionals into three types:

- task-oriented people, motivated by the work they do, who prefer to work alone

- self-oriented people, motivated by personal success and recognition, who prefer to work alone
- interaction-oriented people, motivated by the presence and actions of coworkers, who prefer to work as part of a group.

Group working

Project teams for software development usually have a size that ranges from two to several hundred of people. Large teams are usually split into a number of groups, each one responsible for part of the overall system and which should have no more than 8 – 10 members.

Putting together a group that works effectively is a critical management task [52]. The group should have the right balance of technical skills, experience and personalities, but should also have a team spirit, so that the people involved are motivated by the success of the group as well as by their own personal goals.

Group working is influenced by:

- group composition
- group cohesiveness
- group communications
- group organisation
- working environments.

3.2.2 The People Capability Maturity Model

The P-CMM is a framework for assessing the way organizations manage their staff [53].

The Software Engineering Institute in USA is engaged in a long-term programme of software process improvement. Part of this programme is the CMM for software processes, but also the PCMM has been proposed. PCMM can be used as a framework for improving the way in which an organisation manages its human assets.

PCMM is a five-level model, where the levels are: initial, repeatable, defined, managed, optimizing. The strategic objectives of the PCMM are:

- to improve the capability of software organisations by increasing the capability of their workforce,
- to ensure that software development capability is an attribute of the organisation rather than of a few individuals,
- to align the motivation of individuals with that of the organisation,
- to retain valuable human assets within the organisation.

So, PCMM is a practical tool for improving the management of people in an organisation because it provides a framework for motivating, recognising, standardising and improving good practice.

History of the People CMM

In the early 1980s Watts Humprey noticed that the quality of a software product was tied to the quality of the process that produced it. In order to improve the development processes, Humprey wanted to install a Shewart-Deming improvement cycle into a software organization. What he realized was that a Plan-Do-Check-Act cycle required to be installed in stages, so that the impediments to continuous improvement could be removed. Humphrey's unique insight was that an environment supporting continuous improvement could be created only if implementation problems were eliminated in a specific order. This idea led him and his colleagues at IBM to the development of the concept for a process maturity framework [11].

The staged structure that underlies the maturity framework was first elaborated by Philip Crosby [9]. His original formulation was that any new practice would be adopted through five stages. These stages represented the organization becoming aware of the new practice, learning more about it, trying it in a pilot implementation, deploying it across the organization and achieving mastery in its use.

Humphrey noticed that in the long-term the adoption of practices or technologies was not succeeding because of some problems deeply ingrained in the organizations' culture. So, he realized that he had to formulate an approach that addressed the organization, not just its individual processes. He observed that improved software development practices did not survive unless the organization behaves in a way to support them. Consequently, he designed the process maturity framework.

Through software process assessments, workshops and extensive review, the Software Engineering Institute (SEI) evolved Humphrey's process maturity framework into the Capability Maturity Model for Software (SW-CMM). Later on, through the collaboration with representatives from industry, government, military and academic organizations, the SEI developed an evolutionary model for developing and optimizing employee training and competence in organizations, the so-called People Capability Maturity Model (P-CMM).

Why a CMM for people?

“As other sources of competitive success have become less important, what remains as a crucial differentiating factor is the organization, its employees and how it works” [37]: this quotation is intended to highlight that competent and well-trained employees are the main source of strategic advantage for a company.

However, even if most companies recognize the advantage embodied within talented employees, Curtis et al. [11] point out that those companies often lack a coherent approach to achieve their talent goals and to take in a system of practices.

Practices are really needed, as organizations struggle to deal with recruiting, training and retaining workforce because of labour scarcity, rapidly changing business environment and new working conditions (e.g. life-long employment is no longer the norm).

These needs are exacerbated within the software-development industry. The investment required in knowledgeable people is inherently high because of the constant progress in technologies and programming languages, but also – mainly – because of the growing demand for software opposed to a talent shortage.

As Curtis et al. [11] say, by the mid-1980s the software industry realized that its primary problem was a lack of discipline, both in project management and in software development practices. Since the beginning of then 1990s, the SW-CMM has guided many software organizations in improving their management and development processes. However, software organizations quickly understood that improvements couldn’t come along without significant changes in the way they managed people, but these changes were not fully accounted for in the SW-CMM. That’s where the People CMM came out: the People CMM was designed to increase the capability of the workforce, just as the SW-CMM increased the capability of the organization’s software development processes.

The lack of a coherent approach to manage the issues related to recruiting, training and retaining employees gets along with the idea of this management as “an operational matter ‘to be left to the Human Resources function’” [11]. An integrated human capital management perspective should be taken instead.

Workforce practices are usually considered integral to a total quality management (TQM) program and are included as criteria in quality models such as the Malcolm Baldrige National Quality Award or the European Foundation for Quality Management [11].

Along with the coherent approach and the integrated human capital management perspective, management commitment and a piecemeal approach to adoption are required so that the well-known workforce practices can be properly implemented and can actually produce the already demonstrated benefits.

The People CMM is designed to allow software organizations to integrate workforce improvement with software process improvement programs based on the SW-CMM.

Benefits of a P-CMM

The People CMM is a framework for organizations to focus on the continuous improvement of the management of their workforces. It’s based on the best practices within the fields of human resources, knowledge management and organizational development.

In particular, the People CMM “helps organizations

- characterize the maturity of their workforce practices,
- establish a program of continuous workforce development,
- set priorities for improvement actions (through the staged framework),
- integrate workforce development with process improvement, and
- establish a culture of excellence” [73].

Thus, the People CMM provides organizations with guidance on how to gain control of their processes for managing and developing their workforce through assessing their current maturity and providing a focused set of practices. Those practices are chosen because they have been proved having significant impact on individual, team, and organizational performance.

What is P-CMM?

The People Capability Maturity Model (P-CMM) is a set of practices that constitute a model for organizations to manage their human capital. It consists of five maturity levels (also called evolutionary stages) through which an organization's workforce practices, processes and the organization's culture evolve.

It's worth to notice that although many process standards can transform an organization's culture, few include a roadmap for implementation: that's why organizations often fail to implement the standard effectively. The People Capability Maturity Model, instead, provides a roadmap for continuously improving the capability of an organization's workforce.

The improvement path described by P-CMM is evolutionary and brings the organization's workforce practices from ad hoc, inconsistently performed to a mature whole that continuously elevates the workforce capability.

The P-CMM philosophy can be summarized in ten principles [75]:

1. In mature organizations, workforce capability is directly related to business performance.

2. Workforce capability is a competitive issue and a source of strategic advantage.

3. Workforce capability must be defined in relation to the organization's strategic business objectives.

4. Knowledge-intense work shifts the focus from job elements to workforce competencies.

5. Capability can be measured and improved at multiple levels, including individuals, workgroups, workforce competencies, and the organization.

6. An organization should invest in improving the capability of those workforce competencies that are critical to its core competency as a business.

7. Operational management is responsible for the capability of the workforce.

8. The improvement of workforce capability can be pursued as a process composed from proven practices and procedures.

9. The organization is responsible for providing improvement opportunities, while individuals are responsible for taking advantage of them.

10. Since technologies and organizational forms evolve rapidly, organizations must continually evolve their workforce practices and develop new workforce competencies.

The People CMM, as all the CMMs, is based on five levels of maturity; a maturity level is a set of related practices for some process areas that improve the organization's overall performance. An organization achieves a new level of maturity when a system of practices has been implemented and new results are achieved. The practices at each level of maturity prepare the organization to adopt practices at the next level.

Levels of maturity

The People CMM, which applies the principles of the maturity framework to the domain of workforce practices, consists of five maturity levels that lay successive foundations for continuously improving talent, developing effective teams and successfully managing the human assets of the organization.

Each maturity level represents a different level of organizational capability to manage and develop the workforce: it’s the level of knowledge, skills and process abilities available for performing an organization’s business activities. It aims to produce a transformation in the organization’s culture through providing effective practices to attract, develop, organize, motivate and retain the workforce.

Except for Level 1, each maturity level is characterized by a set of practices that belong to several key process areas of workforce management [73].

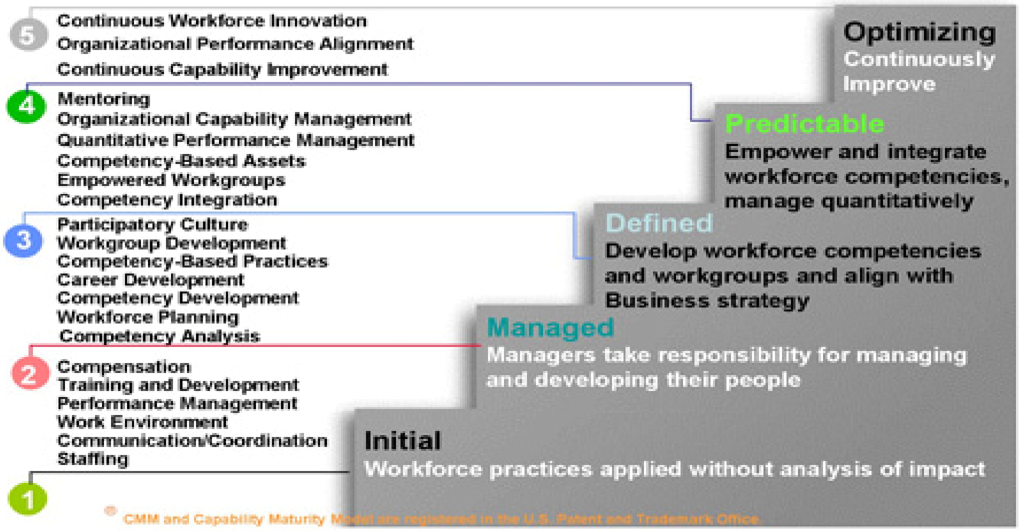

5
Continuous Workforce Innovation
Organizational Performance Alignment
Continuous Capability Improvement
Optimizing
Continuously Improve
4
Mentoring
Organizational Capability Management
Quantitative Performance Management
Competency-Based Assets
Empowered Workgroups
Competency Integration
Predictable
Empower and integrate workforce competencies, manage quantitatively
3
Participatory Culture
Workgroup Development
Competency-Based Practices
Career Development
Competency Development
Workforce Planning
Competency Analysis
Defined
Develop workforce competencies and workgroups and align with Business strategy
2
Compensation
Training and Development
Performance Management
Work Environment
Communication/Coordination
Staffing
Managed
Managers take responsibility for managing and developing their people
1
Initial
Workforce practices applied without analysis of impact
® CMM and Capability Maturity Model are registered in the U.S. Patent and Trademark Office.

Fig. 3.4 P-CMM levels of maturity [73]

<u>First - The Initial level</u>

At this level, the organization has no consistent way of performing its work: processes are ad hoc, results depend on the skills of exceptional individuals and on excessive overtime.

Only repeated practices can be improved: the impediments to repeatability (such as a committed delivery date, uncontrolled requirement changes and so on) must be removed in order to allow the repetition of successful software development practices [11].

Organizations at this level also have difficulty retaining talented individuals. The management of human assets depends on the skills of the managers, who often do not share a common vision and whose responsibilities are rarely clarified (so recruiting and identifying training needs, for example, are displaced to HR or other staff groups). So, when the organization fails in managing and developing its workforce, career-oriented people start under-performing, the loyalty declines and individuals carry on their skills' development in order to pursue career opportunities elsewhere [12].

To solve this problem, managers have to take responsibility for the capability and development of their subordinates.

Second - The Managed level

At this level, the workforce practices implemented focus on activities at the unit level, such as staffing, coordinating commitments, providing resources, managing performance, developing skills and making compensation decisions. In fact, without a basic management control of daily work no organization-wide practice can be successfully deployed. People should be enabled to repeat successful practices: to enable this repeatability, mangers must get control of commitments and baselines. Discipline has to be established [11].

Focusing at the unit level first also establishes a foundation in managing performance that can be used for practices at higher levels.

At this level of maturity, managers are vigilant for problems that hinder performance in their units, such as work overload, environmental distractions, unclear performance objectives or feedback, lack of relevant knowledge or skills, poor communication and low morale [12].

They accept personal responsibility so that workforce practices are implemented effectively in their units. Their focus on managing individual performance and coordinating individual contributions leads to effective unit performances.

The organization which reaches this level of maturity is characterized by the capability of units to meet commitments, capability achieved by ensuring that people have the skills needed to perform their assigned work and that performance is regularly discussed to identify actions that can improve it [12].

Voluntary turnover starts to reduce, because people begin to see a more rational work environment emerging in their unit and their motivation to stay is enhanced [12].

→ The key process areas at Level 2 focus on instilling basic discipline into workforce activities. They are: Work Environment, Communications, Staffing, Performance Management, Training, Compensation.

Third - The Defined level

At this level, the organization identifies its best practices and integrates them into a common process, an organization-wide infrastructure. These practices are also documented and their measures are defined and collected into a repository. Best practices at this level allow to develop those competencies that are critical enablers of business strategy [11].

The concept of workforce competencies implemented in the People CMM differs from the concept of core competency popularized by Prahalad and Hamel [39]. Core competency refers to an organization's combination of technology and production skills that create its products and services and provide its competitive advantage in the marketplace. In P-CMM workforce competencies reside one level of abstraction below an organization's core competency: they represents an integration of the knowledge, skills and process abilities required to perform some of the business activities that contribute to an organization's core competency [12].

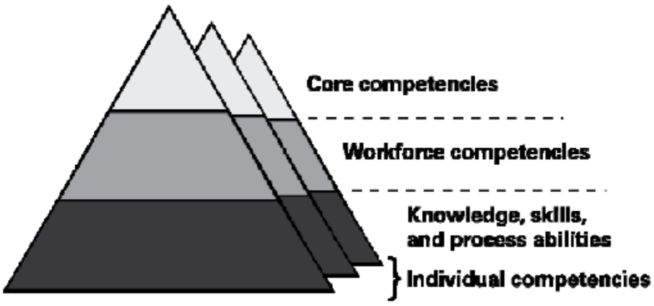


Fig. 3.5 Hierarchy of competency [12]

Thus, the organization-wide infrastructure is an element of the strategic business plan, which identifies the actions that have to be taken for developing the level of talent needed, and must evolve as business conditions and technologies change.

Competency-based processes form a basis for defining workgroup roles and operating processes: workgroups can apply these standard competency-based processes rather than relying only on the interpersonal coordination skills developed at maturity level 2.

→ The key process areas at Level 3 address issues surrounding the identification of the organization's primary competencies and aligning its people management activities with them. They are: Knowledge and Skills Analysis, Workforce Planning, Competency Development, Career Development, Competency-Based Practices, Participatory Culture.

Fourth - The Predictable level

Data describes the organization’s performance: through these data, the processes can be managed quantitatively and the performance becomes much more predictable [11].

There are at least three ways in which the framework of workforce competencies enables the organization to more fully use the capabilities of its workforce [12]:

- When competent people perform their assignments using proven competency-based processes, management trusts the results they produce and this trust enables the organization to preserve the results of performing competency-based processes and develop them as organizational assets. Then learning spreads rapidly and productivity rises.

- This trust also gives managers the confidence they need to empower workgroups, to transfer responsibility and authority. The conditions required for empowerment (competent people, effective processes, a participatory environment) are established. Management senses less risk in empowering workgroups and is willing to delegate increasingly greater levels of authority for managing day-to-day operations and for performing some of their own workforce practices. Managers at this level are able to turn their attention to more strategic issues.

- The organization is able to integrate different competency-based processes into a multidisciplinary process that better integrates the work of several workforce competencies. Such multidisciplinary processes have proven to accelerate business results.

The performance of competency-based processes is measured: these measures are used to establish process performance baselines and to assess the need for corrective action. The creation and use of these baselines and associated measures is similar to the methods that underlie Six Sigma programs: they can be used for planning, for targeting improvements and for predicting the organization’s capacity for work.

The quantitative management capabilities implemented at this level provide management with better input for strategic decisions, while encouraging delegation of operational details to people close to the processes [12].

→ The key process areas at Level 4 focus on quantitatively managing organizational growth in people management capabilities and in establishing competency-based teams. They are: Mentoring, Team Building, Team-Based Practices, Organizational Competency Management, Organizational, Performance Alignment.

Fifth - The Optimizing level

The organization uses its profound, quantitative knowledge to make continuous improvements in its processes: benefits and defects are pointed out and change management becomes an ordinary business process to be performed in an orderly way on a regular basis [11].

The organization must be vigilant to ensure that performance at all levels remains aligned with organizational objectives: the process performance data collected across the organization is evaluated to detect instances of misalignment. Corrective actions are taken when necessary.

Everyone strives to improve his own capability and contributes to improvements in the performance of the workgroup, the unit and the organization [12].

→ The key process areas at Level 5 cover the issues that address continuous improvement of methods for developing competency, at both the organizational and the individual level. They are: Personal Competency Development, Coaching, Continuous Workforce Innovation

Themes

Four themes characterize the P-CMM [86]:

- Developing capabilities, which means that training needs are identified, core competencies are developed and individuals establish their own professional development.

- Building teams and culture, meaning that basic communication skills are established and a certain culture based on team building and improvement of team capabilities is developed.

- Motivating and managing performance, which means that basic performance management is established and compensations practices are implemented.

- Shaping the workforce, that is establishing basic staffing practices, developing plans for workforce development, set and track objectives for competencies in the workforce and look for sources of innovation.

Key process areas (KPA) refer to the particular tasks and activities which must be completed in order for an organization to gain maturity and progress towards optimizing their training initiatives. The following matrix identifies the appropriate KPA necessary to address each of the four themes of the P-CMM, and allow the organization to mature [86].

Maturity Levels	*THEME 1:* *Developing capabilities*	*THEME 2:* *Building teams and culture*	*THEME 3:* *Motivating and managing performance*	*THEME 4:* *Shaping the workforce*
5: Optimizing	coaching personal competency development		continuous workforce innovation	
4: Managed	mentoring	team building	organizational performance alignment team-based practices	organizational competency management
3: Defined	competency development knowledge and skills analysis	participatory culture	competency-based practices career development	workforce planning
2: Repeatable	training communication	communication	compensation performance management work environment	staffing
1: Initial				

Tab. 3.1 KPA to address for each P-CMM theme and maturity level [86]

Implementation

The implementation of the People Capability Maturity Model (P-CMM) requires support and approval from the different areas of an organization. It has to be interpreted and customized; no level can be skipped.

To interpret and implement the model in a organization, the following criteria, related to each Key Process Area, may help [86]:

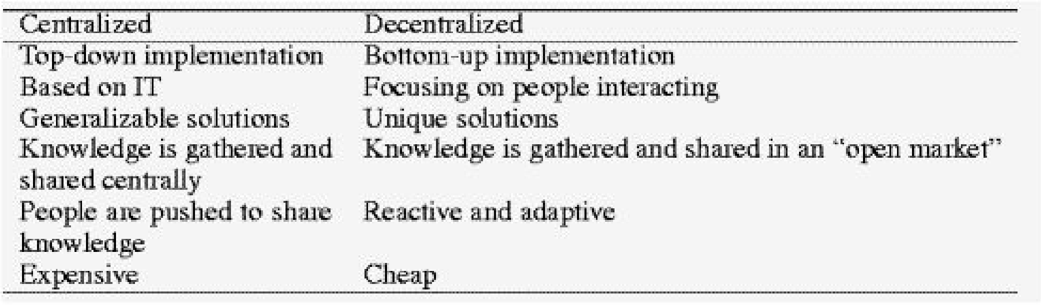
- goals
- commitments to perform
- abilities to perform
- activities performed
- measurement and analysis
- verification of implementation.

3.2.3 A decentralized approach to people management: the broker model

Centralized vs decentralized approaches

From the beginning, the management of human resources has been considered from a top-down perspective. However, as Hellström et al. [24] underline, other factors rather than personality traits and hierarchy are important to the management of people: multiple role-taking, reflexivity, knowledge brokerage and the affecting of initiative, for example. They suggest that decentralized theories of the management of knowledge may be as good as the top-down ones.

Sarvary [41] describes through a table the main characteristics of a centralized versus decentralized approach to knowledge management:



Centralized	Decentralized
Top-down implementation	Bottom-up implementation
Based on IT	Focusing on people interacting
Generalizable solutions	Unique solutions
Knowledge is gathered and shared centrally	Knowledge is gathered and shared in an "open market"
People are pushed to share knowledge	Reactive and adaptive
Expensive	Cheap

Tab. 3.2 Centralized vs decentralized approach to knowledge management [24]

A decentralized approach to knowledge and communication is discussed by Hedlund [23], as he judges preferable a combination of human resources and a lateral, lower/middle communication in the company. He also points out that this approach affects leadership and organizational strategy in a company, as the manager is no longer a resource allocator but a catalyst and the strategy shifts from a semi-independent diversification to a knowledge-based competition.

This approach leads to the conception of the firm as a distributed knowledge system [61], i.e. the knowledge cannot be collected in only one mind, and to the so-called cellular organization [33] where the responsibilities about the organization of the work are disseminated through workers and workers own what they create.

Prusak and Coehn [40] assume that the knowledge within an organization is driven by the same forces that rule the markets for tangible goods. They identify several actors belonging to this particular market of knowledge inside the company: buyers, sellers, brokers; also there is a pricing system (informal or pre-capitalist) and trust is needed as well as a sense of reciprocity and repute .

So, for a decentralized approach to work, a certain kind of knowledge market should be created and knowledge exchanges should be recognized and promoted.

A case study: Knowledge brokerage at EST

Hellström et al. [24] propose a case study, “Knowledge brokerage at EST” that describes a decentralized approach to the management of knowledge work (i.e. how the skilled mental labour leads to commercial products).

EST is a Ericsson company that develops software applications for telephones. One of its goals has been to climb the ISO 9000 and the Capability Maturity Model, considered as an assurance of quality of the software program code. This has led EST to look for a way to coordinate and leverage the company’s competence and knowledge through projects and units: the chosen way was to implement a database called Experience Factory.

The Experience Factory was a tool-centered database for collecting, storing, analyzing and reusing collective experience of software engineers in order to make cost estimations and fault predictions. However, the Experience Factory presented the same problems of already existing databases.

A corridor study and workers interviews brought the team to believe that in the factory approach two important elements of the knowledge exchange process were missing:

- the context of the exchange, which needed to be personal and physical
- the situational nature of the exchange itself, as the exchange is a consequence of spontaneous demands addressed within meetings that lead

to informal ideas.

These elements hinted that knowledge seemed to be exchanged more effectively when two parties where engaged in an activity that required knowledge to solve some kind of problem. The exchange occurred in two primary ways: as flashes and as learning situations.

As the Experience Factory approach was being implemented, some issues arised:

- The users judged the database as too technology-based, administratively-demanding and time consuming.
- The standardization and formalization of the approach was perceived to be an end in itself and the measurements relied on parameters that seemed distant from the core activity.
- The context for evaluating data and information was hard to tell or write.
- The information was found to have a very short life span in software engineering.
- Data on workers' competences became quickly obsolete.
- Knowledge without a recipient was harder to be shared.
- The bonus system didn't encourage cooperation.

To solve these problems, a less formalized approach was suggested, as spontaneous and informal face-to-face meetings were recognized as the most important field for exchange: the Experience Engine was created. This "social engine" aims to recycle the collective experience of the company.

To develop this semi-formalized broker model, an empirical study was carried on. It emerged that a reinforcement was required, so that the process for knowledge exchange would effectively take place: the figure of a knowledge broker was introduced as a catalyst for connecting those in need for knowledge and those who owned it. This role was supposed to be independent: the broker is simply a connector for people to exchange knowledge.

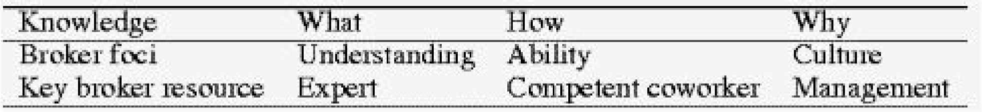
The introduction of the broker had an impact on the number of exchanges between experts and workers and brought a reduction of lead times in projects. It was planned to establish an Intranet site to increase transparency and reduce search costs.

As Hellström et al. [24] say, the successful evolution of a broker model rather than a management system of command control mirrors the prevalence of market over hierarchy. This constitutes a challenge in managing knowledge work, because an equilibrium between these two organizational rationalities should be found.

The role of the broker is delicate and that's why it should be "slightly unofficial" but very visible at the same time. It's important to define its scope, but also what kind of knowledge is being brokered, what processes and capabilities the broker may enable and what are the key resources required.

The broker model is not without restriction and constrains. As described before, there is a trade-off between the efficiency of the internal knowledge market and the organizational hierarchy.

Hellström et al. [24] conclude that the broker has to face different knowledges, focuses and resources, summarized in the following table:



Knowledge	What	How	Why
Broker foci	Understanding	Ability	Culture
Key broker resource	Expert	Competent coworker	Management

Tab. 3.3 Knowledges, focuses and resources a broker has to face [24]

There are three main aspects in the broker model. First, people who need a specific piece of knowledge, "knowledge what", should be connected to its owner, a key resource called expert, in order to lead to a better understanding of what is lacking.

Second, knowledge may concern how to do something, so the aim of the broker would be to create the ability to perform certain tasks though a key resource, a competent coworker.

Third, "knowledge why" may be exchanged too: the focus is on the culture and market and the key resources are managers.

These three types of knowledge broking focuses, when executed, can lead the decentralized knowledge management model to succeed.

3.2.4 The virtual incubator: a network of specialists

The software industry and its evolution

The software industry lies at the heart of the new knowledge-based economy [35] because it provides tools and infrastructures for all the other industries. This industry, despite the presence of some big players, is made up of small companies and entrepreneurs.

About ten years ago, Nowak and Grantham [35] predicted that this industry was expected to evolve from the object-oriented programming paradigm to the component-based software development paradigm where only generic components would be created and purchased, so that business or domain experts would modify them and create specific applications. The effect of this specialization of work and of a greater focus on the areas of expertise would lead to “dramatic increases in the quality, maintainability and flexibility of software while reducing its cost, development time and complexity”.

Globalization was also expected to wrap around the software industry. Thanks to the Internet growth, all producers would find an easy access to mass markets. On the other hand, global scales of economy and brands would be possible to be created by large companies. As Nowak and Grantham [35] underlined, the globalization of the world economy would also lead the members of the fragmented software industry to form strategic partnerships to fight the constant demand for technological innovation and for shorter product development cycles in order to remain competitive.

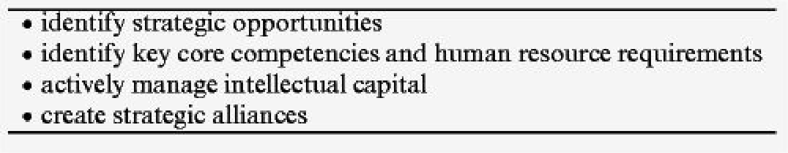
Requirements for software start-ups

In 1997 the California Trade and Commerce Agency sponsored a collection of interviews with software industry executives, professional organizations and regional industry councils in order to study the main requirements of software start-ups. This study pointed out three critical needs: the access to low cost infrastructure resources, the access to adequate management skills and knowledge and the access to business networking resources for marketing. What appeared to be lacking was knowledge about business planning, competitive assessment, marketing, sales, financial planning and analysis, human resources.

The critical human resources for a start-up to be successful were not provided by either the private or the public sector. Nowak and Grantham [35] believed that a cooperation between universities, privates and the public sector was needed, so they proposed a new model to facilitate start-ups to succeed and business network to form.

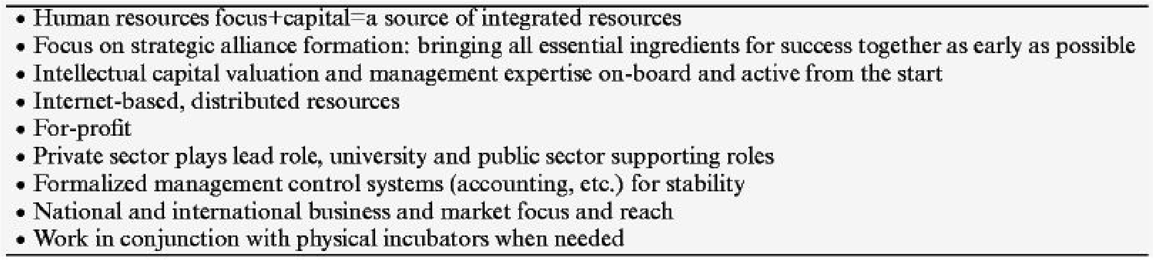
The virtual incubator

Basically the new model needs to provide a structure, the so-called virtual incubator, to easily access information, experience and resources: the focus has to be on wealth creation and the tool is a virtual “network of innovation” which collects excellence in the technical and managerial fields. Key steps of the model’s methodology and main characteristics of the virtual incubator are shown in the tables.



- identify strategic opportunities
- identify key core competencies and human resource requirements
- actively manage intellectual capital
- create strategic alliances

Tab. 3.4 Key steps of the virtual incubator methodology [35]



- Human resources focus+capital=a source of integrated resources
- Focus on strategic alliance formation: bringing all essential ingredients for success together as early as possible
- Intellectual capital valuation and management expertise on-board and active from the start
- Internet-based, distributed resources
- For-profit
- Private sector plays lead role, university and public sector supporting roles
- Formalized management control systems (accounting, etc.) for stability
- National and international business and market focus and reach
- Work in conjunction with physical incubators when needed

Tab. 3.5 Main characteristics of the virtual incubator [35]

As a conclusion, Nowak and Grantham [35] believed that in the future successful models would include "small-scale networks of interlocked specialists coming together on a temporary basis to approach a focused market or software project", underlining once again the importance of the human element.

4 Software quality

4.1 What does it mean to manage software quality?

4.1.1 Managing software quality

Quality assurance

Software quality management is about defining the software development processes to be used and the standard to be applied, but also is about checking that the planned processes have been followed, the project outputs are conformant with the standards.

While in the manufacturing industry the terms “quality assurance” and “quality control” mean the definition of processes and standards that should lead to high-quality products and their application, in the software industry “quality assurance” means the verification and validation and the processes of checking that quality procedures have been correctly applied and “quality control” is not so used [54].

Thus, software quality assurance (SQA) is the monitoring of the software engineering processes such as requirements definition, software design, coding, source code control, code reviews, change management, configuration management, testing, release management and product integration. SQA aims to ensure the quality of these processes though checking their conformance to a standard, such as ISO 9000, or model, as CMM.

Quality management: the team

The team in charge of quality management should be independent, not tied to any other group (such as development groups): it’s the only way to ensure that the goals of quality are not sacrificed because of budget or schedule problems. However, in small companies this hardly happens.

Quality plan

According to Humphrey [27] a quality plan should be made. It should include: product introduction (description of the product, the market and the quality expectations), product plans (release dates, responsibilities, distribution plans), process descriptions, quality goals, risks and risk management [54]. However, there is much more to quality management than what’s included in the plan:

standards and processes are important, but a “quality culture” should be set up too.

4.1.2 Defining quality: software quality factors

Quality is subjective

As hinted before, software quality is not exactly the same of quality in manufacturing, because specifications are never complete and unambiguous, requirements come from different stakeholders and they may not include everything and also certain quality characteristics are difficult, where not impossible, to measure [55].

It’s worth notice that these attributes cannot be all optimized at the same time: the quality plan should define the most important attributes for the software which is being developed.

Thus, the quality of a software product is often subjective, as it is determined not only by the customer’s functional requirements, but also by several desirable non-functional factors.

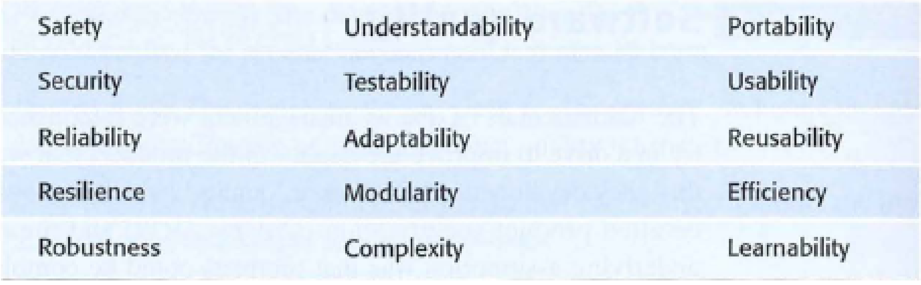
Software quality factors

The main non-functional factors that impact on software quality are the following [81]:

- Understandability of all the design and user documentation.
- Completeness of the code: all the constituent parts should be present and all the required input data available.
- Conciseness: since the memory capacity is limited, lines of code should be kept to a minimum through avoiding redundant information or processing.
- Portability on different computer configurations, either different hardware and operating systems.
- Consistency in notation, symbology and terminology.
- Maintainability: updates should be facilitated through a good documentation, resources should be spared.
- Testability: evaluation of performance should be easy to make.

- Usability: user interface should be friendly.
- Reliability, as described previously, of the product to perform correctly over a period of time.
- Efficiency in terms of use of resources such as memory, space and processor utilization, network bandwidth, time.
- Security of data against unauthorized access or interferences.

A similar suggestion comes from Bohem et al., as they list 15 important software quality attributes that overlap the previous ones [5]:



Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Tab. 4.1 Software quality attributes [55]

4.1.3 Measuring software quality

Measuring the software quality attributes is difficult

Measures can be approached differently. Qualitative and quantitative measures may be needed. However, as Kaner [72] and Hoffman [71] point out, it’s difficult to measure what we truly want to measure well.

As already said before, it’s difficult to make direct measurements of many of the software quality attributes, because many of them are affected by subjective factors. So, some internal characteristics of the software have to be accurately measured and it has to be assumed that there is an understood, validated and modelled relationship between them and the attributes that are needed to be assessed. Examples are given in the following picture [58].

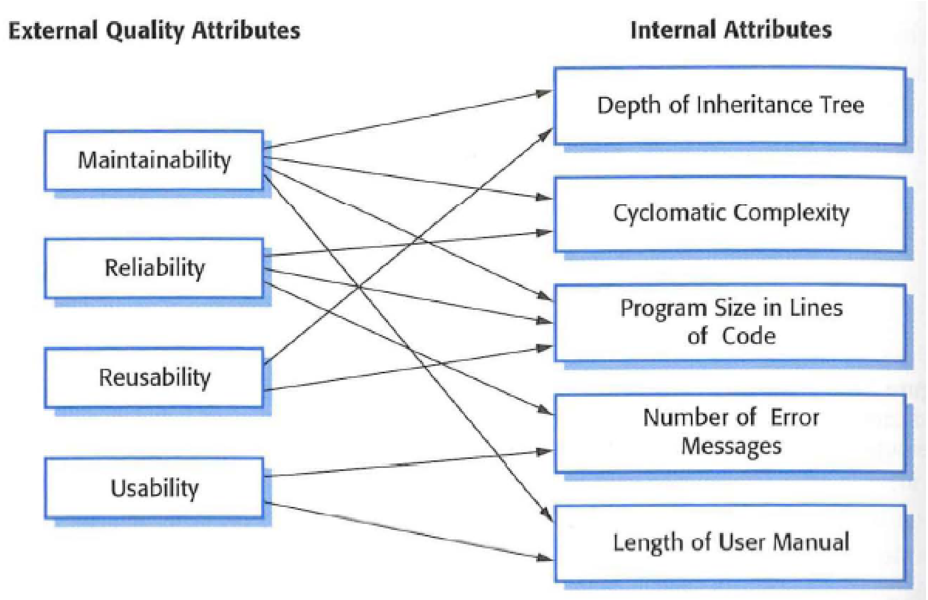


Fig. 4.1 Relationships between internal and external attributes [58]

Measurements and metrics

Software measurement involves linking a numeric value or a profile to an attribute of a software component, system or process. The quality of the software and the effectiveness of the related processes can be evaluated through comparing these values to each other and to the chosen standards. Possible improvements are also highlighted.

Software metrics are the characteristics of a software system, system documentation or development process that can be objectively measured. They may be control metrics, which support the process management, or predictor metrics, which help to predict some characteristics of the software.

These metrics influence the management decision making, even if they won't necessarily convince the decision makers.

Number of faults

A common metric used is the number of faults encountered in the software: the more this number is little, the higher the quality is supposed to be. Still, this number should be put in the right context, meaning that aspects like size and complexity of the software, severity of the discovered faults and the incidence of users affected by a certain fault should be considered.

Also, another difficulty comes from software being a human creation: software programmers and testers may be tempted to trick the measurements.

Measurable attributes

The software quality factors as previously described cannot be measured because of their vague definitions. So, measurable attributes related to the software quality factors must be identified in order to quantify the non-functional requirements.

Examples of relevant questions to be asked for each software quality factor are reported [81]:

Software quality factor	**Relevant questions for measurement**
Understandability	Are variable names descriptive of the physical or functional property represented? Do uniquely recognizable functions contain adequate comments so that their purpose is clear? Are deviations from forward logical flow adequately commented? Are all elements of an array functionally related?
Completeness	Are all necessary components available? Does any process fail for lack of resources or programming? Are all potential pathways through the code accounted for, including proper error handling?
Conciseness	Is all code reachable? Is any code redundant? How many statements within loops could be placed outside the loop, thus reducing computation time? Are branch decisions too complex?
Portability	Does the program depend upon system or library routines unique to a particular installation? Have machine-dependent statements been flagged and commented? Has dependency on internal bit representation of alphanumeric or special characters been avoided? How much effort would be required to transfer the program from one hardware/software system or environment to another?
Consistency	Is one variable name used to represent different logical or physical entities in the program? Does the program contain only one representation for any given physical or mathematical constant? Are functionally similar arithmetic expressions similarly constructed? Is a consistent scheme used for indentation, nomenclature, the color palette, fonts and other visual elements?
Maintainability	Has some memory capacity been reserved for future expansion? Is the design cohesive—i.e., does each module have distinct, recognizable functionality? Does the software allow for a change in data structures (object-oriented designs are more likely to allow for this)? If the code is procedure-based (rather than object-oriented), is a change likely to require restructuring the main program, or just a module?
Testability	Are complex structures employed in the code? Does the detailed design contain clear pseudo-code? Is the pseudo-code at a higher

	level of abstraction than the code? If tasking is used in concurrent designs, are schemes available for providing adequate test cases?
Usability	Is a GUI used? Is there adequate on-line help? Is a user manual provided? Are meaningful error messages provided? Is the user interface intuitive (self-explanatory/self-documenting)? Is it easy to perform simple operations? Is it feasible to perform complex operations? Does the software give sensible error messages? Do widgets behave as expected? Is the software well documented? Is the user interface responsive or too slow? Also, the availability of (free or paid) support may factor into the usability of the software.
Reliability	Are loop indexes range-tested? Is input data checked for range errors? Is divide-by-zero avoided? Is exception handling provided? It is the probability that the software performs its intended functions correctly in a specified period of time under stated operation conditions, but there could also be a problem with the requirement document...
Efficiency	Have functions been optimized for speed? Have repeatedly used blocks of code been formed into subroutines? Has the program been checked for memory leaks or overflow errors?
Security	Does the software protect itself and its data against unauthorized access and use? Does it allow its operator to enforce security policies? Are security mechanisms appropriate, adequate and correctly implemented? Can the software withstand attacks that can be anticipated in its intended environment?

Tab. 4.2 Examples of questions for the measurement of software quality factors [81]

All the collected data should always be maintained, because it’s an organizational resource that allows significant comparisons, but, most important, identify those relationships between internal characteristics and quality attributes.

Basili and Weiss [2] propose a data collection methodology that consists of six steps. Feedback and iteration may occur. The six steps are:

1. Establish the goal of the data collection
2. Develop a list of questions of interest
3. Establish data categories
4. Design and test data collection forms
5. Collect and validate data
6. Analyze data.

4.2 Different approaches to quality

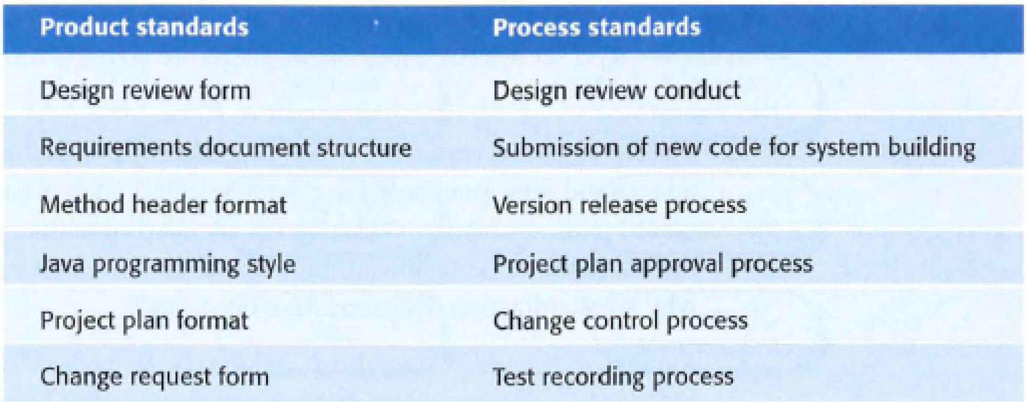
Even though the term quality seems self-explanatory, quality can be approached in several different ways: according to standard definitions, user's perception, customer satisfaction, software lifecycle, etc.

4.2.1 Software standards

Product and process standards

The importance of software standards in the software quality management relies on three reasons: standards capture knowledge and best practices, provide a framework for defining quality and assist continuity within the organization.

There are two main types of standard: product and process standards. An example from Sommerville [56] is:



Product standards	Process standards
Design review form	Design review conduct
Requirements document structure	Submission of new code for system building
Method header format	Version release process
Java programming style	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

Tab. 4.3 Product and process standards [56]

The quality assurance team should collect these standards, which can derive from national and international standards, in a handbook. To encourage the use of standards, managers should ensure that software engineers are involved in the selection of product standards, that the standards are updated and reflect the last technologies and that software tools to support standards are provided [56].

Reviews and inspections

The activities of review and inspect aim to check the quality of the software product, so the software and its documentation are examined to find any error, omission or misalignment to standards. The purpose, of course, is to improve the software quality.

The review is made up of three steps [57]: the preparatory activities, such as review planning and preparation, the review meeting itself and the post-review activities that involve fixing software bugs, rewriting documents, make adjustments in order to respect the standards.

The inspections are “peer reviews” where the team members look for bugs in the software developed. Test cases are an effective way to inspect. Also a checklist of the most common errors should be used.

However, reviews and inspections can take place effectively only whether there is a supportive culture that doesn’t blame the individual when errors are discovered.

ISO 9001 – processes and quality

The ISO 9000 standards define quality as the conformity to requirements specification. ISO 9001 is a set of standards that can be used by organizations that design, develop and maintain products such as software. It identifies some general quality principles, describes quality processes in general and lists the organizational standards and procedures to define. To be ISO 9001 conformant, an organization has to document the relationships between its processes and the core processes identified during the 2000 revision, reported in the following picture [57].

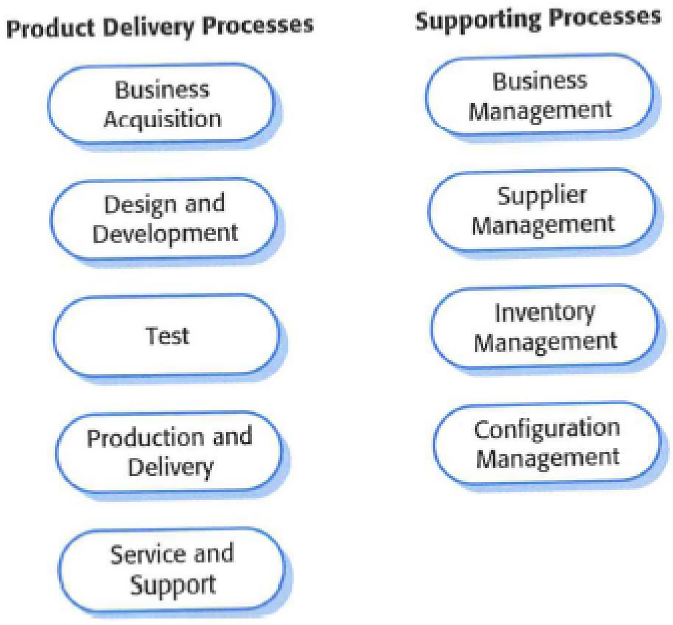


Fig. 4.2 ISO 9001 core processes [56]

The ISO 9001 standard pushes the organization to document its processes through a quality manual. This kind of handbook is used to develop those quality plans that support the quality management.

ISO 9126 – product quality

ISO/IEC 9126 is an international standard for the evaluation of software which focuses on the so-called product quality, the characteristic of the product determined by the presence of some measurable product attributes [16].

The standard is divided into four parts [79] which addresses, respectively, the following subjects:

- quality model
- external metrics, which are measurable within the running of the software
- internal metrics (also called static measures), which do not depend on software execution
- quality in use metrics, which are available when the product is used in real conditions.

Ideally, the internal quality determines the external quality and external quality determines quality in use.

To evaluate software, characteristics that are relevant for quality need to be selected; they depend on the context, the type of user (end user or a person maintaining or porting the software, for example), etc.

The ISO/IEC 9126 software quality model identifies six main quality characteristics: functionality, reliability, usability, efficiency, maintainability and portability. These characteristics are broken down into sub-characteristics and each sub-characteristic is further divided into attributes. Attributes are not defined in the standard, as they vary between different software products. The following table collects the six characteristics, several sub-characteristics and their definitions [4][85][79].

Functionality	Functionality is essential: without it, none of the other characteristics can be accomplished. It tells whether a set of functions and their specified properties, which satisfy stated or implied needs, exist.	Suitability	Appropriateness (to specification) of the functions of the software.
		Accuracy	Correctness of the functions.
		Interoperability	Ability of a software component to interact with other components or systems.

		Compliance	Conformity to certain industry or government laws and guidelines.
		Security	Regulation of the authorizations to access the software.
Reliability	It defines the capability of the system to work and maintain a certain level of performance under defined conditions for a defined period of time.	Maturity	Frequency of failure of the software.
		Fault tolerance	Ability of software to withstand (and recover) from component or environmental failure.
		Recoverability	Ability to bring back a failed system to full operation, including data and network connections.
Usability	It refers to the ease of use for a given function.	Understandability	Determines the ease of which the systems functions can be understood by the user
		Learnability	Learning effort for different users, i.e. novice, expert, casual etc.
		Operability	Ability of the software to be easily operated by a given user in a given environment.
Efficiency	This characteristic is concerned with the resources used when providing the required functionality and level of performance.	Time behavior	Response times for a given transaction rate.
		Resource behavior	Resources used, i.e. memory, CPU, disk and network usage.
Maintainability	It’s the ability to identify and fix a fault within a software component or to make specified changes. It is also referred as supportability.	Analyzability	Ability to identify the root causing a failure within the software.
		Changeability	Amount of effort to change a system.
		Stability	Sensitivity to change.
		Testability	Effort needed to verify (test) a system change.

Portability	This characteristic refers to how well the software can stand changes in its environment or its requirements.	Adaptability	Ability of the system to change to new specifications or operating environments.
		Installability	Effort required to install the software.
		Conformance / Co-existence	Conformity to a different operating system, application server, etc.
		Replaceability	How easy is it to exchange a given software component within a specified environment.

Tab. 4.4 ISO/IEC 9126 quality characteristics, subcharacteristics and ther definitions

4.2.2 Customer satisfaction

Software specification vs customer satisfaction

According to Denning [13], the currently employed methodologies for software quality are based on the idea that quality is strongly related to software specification and can be achieved through a four stages process:

1. Clearly define the requirements.
2. Construct a formal specification based on the requirements.
3. Derive programs from the specification.
4. Demonstrate that the implement programs meet the specification.

Guidelines are readability, modularity, modifiability, style, adequacy of comments and good tests.

However, software quality can be analyzed also from the point of view of customer’s satisfaction: as Denning [13] points out, “by making concerns of customer satisfactions central among the criteria for judging software, new actions will appear for making reliable and dependable software […]. The greater the level of satisfaction, the more likely the customer is to say the software is good quality and is dependable”.

Levels of satisfaction

As a consequence of this approach, the software design never ends until the customer declares his satisfaction with the software delivered. Denning distinguishes three levels of customer satisfaction [13]:

- All basic promises are fulfilled.
- No negative consequences are produced, what the customer actually wants (and not only what he has asked for) is delivered.
- The customer is delighted, what is produced is beyond his expectations.

Denning [13] observes that there might be a difference between the language of the software designer and the language of the customer, so specifications may not reflect exactly what happens in business processes; specifications should be used to clarify the interactions between subsystems and not between systems and users.

He also lists several negative consequences that can arise: collective phenomena (i.e. when a large number of programs interact), computer hardware failures, mistakes by the users, unforeseen situations that arise, changing of users' expectations. It's a duty of a good software designer to prevent the system from undergoing these problems through including useful functions even if not explicitly required and continuing to work with the customer even after the software installation.

About delight, Denning [13] points out that it's ephemeral when based only on the software: a relationship between customer and performer is what actually allows delight to arise. Program correctness is essential, but quality and dependability of the software are deeply related to customer satisfaction and this requires the software developer to be available and to understand his customer.

4.2.3 User's perception

Perceived quality and quality in use

Quality is strongly related to the perception of the user: Garvin [16] defines the user perceived quality as the combination of product attributes that provide satisfaction to a specific user. Of course it's a subjective judgment, but products have quality only with relation to their intended purpose [4].

Both Garvin and Bevan emphasize quality being related to particular needs in a particular situation: that's how quality in use is defined. As the quality of a product is strictly linked to its purpose, quality in use can be defined as “the effectiveness, efficiency and satisfaction with which specified users can achieve specified goals in specified environments”[4]. It’s worth notice that, according to this definition of quality, the requirements are no longer specified in terms of attributes but as performances.

User centred design

Since the user’s perspective is so important, Bevan [4] suggests a way to achieve quality through considering this perspective: he refers to the ISO 13407 standard. ISO 13407 provides guidance on achieving quality in use through user centred design activities, i.e. activities aim to enhance effectiveness and efficiency in working conditions.

Within each stage of the lifecycle of a project, four activities need to be executed:

1. The context of use has to be understood and specified in terms of characteristics of the intended users, tasks the users are to perform and environment in which the users are to use the product.

2. The user and the organizational requirements have to be specified: explicit statements about usability of the human-computer interface and of the quality in use have to be provided.

3. Design solutions have to be produced, considering the state of the art and the experience and knowledge of the participants.

4. Design has to be evaluated against requirements, so that feedback is provided, user and organizational objectives are assessed to have been achieved or not and the long-term use of the product can be monitored.

The context of use

Quality in use is measured in order to ensure that the software achieves its goals and meets needs in a certain context. The context of use, which might already exist or can be an intended context, is made up of users, tasks, equipment (hardware, software and materials) and the social and physical environment. Measures of quality in use for the intended context are:

- Effectiveness: user’s goals are related to the accuracy and completeness with which they can be achieved

- Efficiency: the level of effectiveness achieved is related to the consume of resources (such as time, effort, materials, money).

- Satisfaction: the comfort and acceptability of the use.

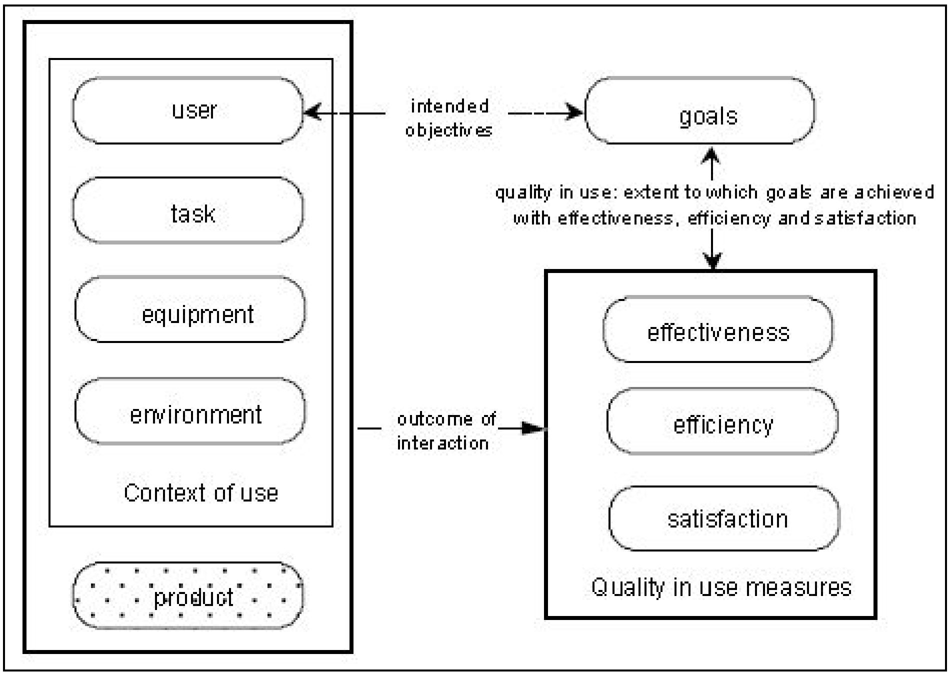


Fig. 4.3 The context of use and quality in use measures [4]

The MUSiC methods: a tool to assess the context of use

Valuable tools to measure effectiveness and efficiency of the system in use are provided by the MUSiC methods. MUSiC stands for Measuring the Usability of Systems in Context. These methods evaluate the extent to which specific task goals are achieved and the time required for this as well as the time spent unproductively because of some problems and diagnostic data about these problems.

Effectiveness is measured as a percentage of the product of the quantity of the task the user completes and the quality of the goals the user achieves.

$$Task\ Effectiveness = \frac{1}{100}(Quantity\ \times\ Quality)\%$$

Efficiency is measured as the ratio of effectiveness to the expenditure of resources (e.g. time required to execute a task).

$$Temporal\ Efficiency = \frac{Effectiveness}{TaskTime}$$

Satisfaction can also be measured. Bevan [4] says that satisfaction is composed of comfort, the overall emotional response to the use of the system, and acceptability, the overall attitude of the user towards the system.

Satisfaction can be measured by attitude rating scales as SUMI (Software Usability Measurement Inventory, is a 50-item standardized questionnaire from the MUSiC project), but can also be assessed indirectly through analyzing positive and negative comments during use, absenteeism, health problem reports, etc.

4.2.4 Software lifecycle and software processes

The quality of the process affects the quality of the product

An assumption in software quality management, derived from manufacturing, is that the quality of software (the product) is directly related to the quality of the software development (the processes). However, the relationship between product quality and process quality is more complex than in manufacturing: as Sommerville [55] says, the influence of individual skills and experience here is significant.

Quality of design and quality of conformance

Software quality concerns two main aspects:

- Quality of design, that concerns implementation, i.e. how well the software is designed; standards guide the developer in software engineering.

  For what concerns the quality of design, the source code quality is affected by readability, ease of maintenance, testing, debugging, modification and portability, low complexity, low resource consumption (memory, CPU), number of compilation warnings, robust input validation and error handling.

- Quality of conformance, that means how well the software conforms to the design, how valid the requirements are in creating a worthwhile product; requirements (even the implicit ones) are the foundations for quality to be measured.

  The quality of the software product, intended as conformance to requirements and specification, is commonly related to reliability. Other characteristics that affect the software product quality are scalability, correctness, absence of bugs, fault-tolerance, extensibility, maintainability and documentation [79].

Software reliability is defined as "the probability of failure-free operation of a computer program in a specified environment for a specified time" [34]. The more critical the application of the software to economic and production processes, the more important is to assess the software's reliability.

Software reliability is objective and can be estimated through measured criteria called software metrics. The main problem in measuring it, tough, is the difficulty of determining exactly in advance how the software is intended to operate: most modern software perform a work which a human could never perform and that may be difficult to understand. Moreover, software lack of qualities such as adaptability, general-purpose knowledge, a sense of conceptual and functional context and common sense, so the expected outcome and the actual outcome may differ because no software can provide those qualities [81].

Software reliability can be improved at different stages of a program's development: requirements, design, programming, build and deployment, testing and runtime evaluation.

Requirements	Defining a program’s desired behavior (i.e. identifying all the requirements) in advance is attractive but impossible. A formalized approach to the process of defining requirements may be helpful.
Design	Design aims to specify how the program should do what is meant to do. Usually the problem is split in sub-problems, typically concerning architecture, overall program concept and structure, coding. Constraints and a program template are provided. Design should be independent of the chosen implementation language.
Programming	The chosen language should enable a better understanding of the program’s overall structure and functionality and lead to a reduction in the number of errors.
Software build and deployment	The software build is the translation operated by a compiler from source code to a form that can be executed by a computer: it’s a critical phase for software quality because if the generated files are incorrect the software will fail. The software deployment is the transfer of the software to the runtime area (i.e. the application server) and it’s also critical because it involves the definition of some technical parameters, which need to be correctly set.
Testing	When done correctly, testing can increase the overall quality of conformance. Tests are: unit testing, functional testing, regression testing, performance testing, failover testing, usability testing.
Runtime	Performance and interoperability with other code or particular hardware configurations are evaluated.

Tab. 4.5 Software reliability can be improved at each stage of software development

Software lifecycle: performances and attributes

Since requirements are linked to performances, quality can be analyzed from the point of view of the software lifecycle, i.e. where performances can be detected [4].

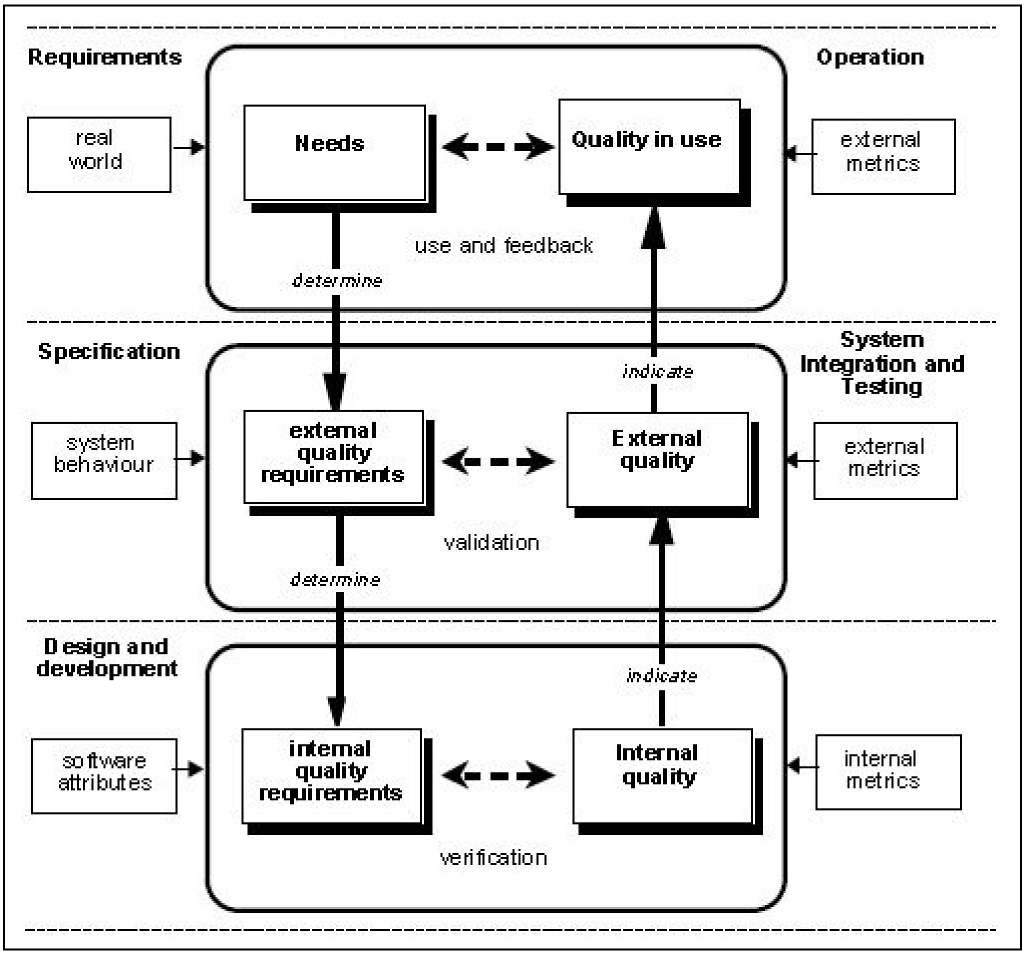


Fig. 4.4 Quality in terms of performances within the software lifecycle [4]

The picture shows the true connection between internal software attributes and quality in use: quality in use is the main goal, software product quality – that is described by software attributes – is the way of achieving it.

The users' needs are translated into a set of requirements for the behavior of the software when in use: to meet these requirements, some software attributes must be present.

The internal quality, i.e. whether the internal requirements have been met, can be checked through internal metrics. On the other hand, the external quality can be assessed only for a complete, working system, so the expected environment and conditions of use have to be emulated. The assessment of external quality can be used in order to provide a feedback, supporting a continual improvement process.

The relationship between internal and external measures and attributes and the quality in use is described in the following picture [4]:

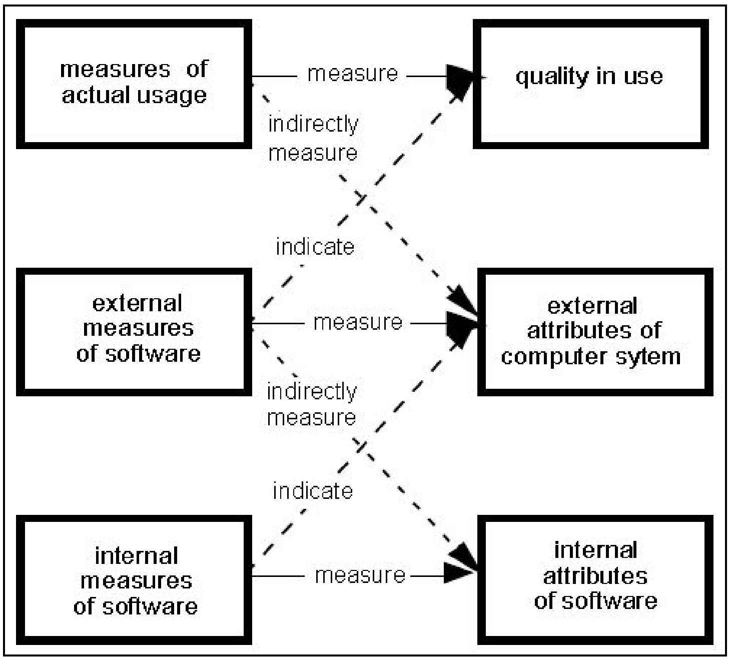

measures of actual usage
measure
quality in use
indirectly measure
indicate
external measures of software
measure
external attributes of computer sytem
indirectly measure
indicate
internal measures of software
measure
internal attributes of software

Fig. 4.5 The relationship between quality in use and interna and external measures and attributes [4]

As Bevan [4] points out, it’s important that internal software quality attributes are directly related to external quality requirements, so that the quality of the software product can be assessed against the quality needs for the software in use.

Quality can be measured during the development process: this may be part of a quality plan. The picture describes the quality plan according to ISO 9000 [4].

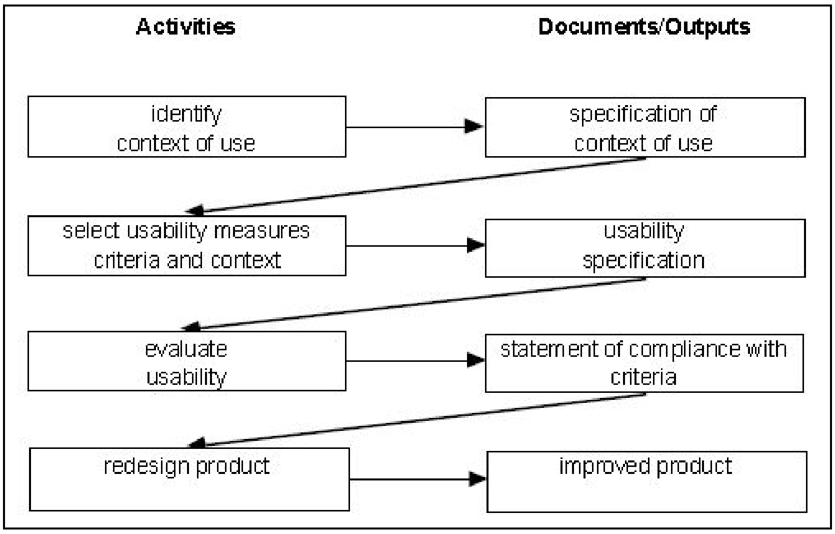


Fig. 4.6 ISO 9000 Quality plan [4]

4.3 How do software people affect software quality?

4.3.1 Software team skills on software product quality

Quality of people is a driver for product quality

As Vinod et al. [63] say, one of the first actions a software development project manager should undertake for his project to be successful is to have skilled and experienced team members.

Even if the relationship between team skills and software quality may be intuitive, there are very few software engineering models based on it. In fact, the primary focus of most software quality models is on processes.

It's certainly true that the quality of the software product depends on the quality of its development process, but skills and experiences of the software development team members should be considered as well a driving factor in software product quality. Authors like Bach [1] go further, insisting that the quality of people is not just a driver, but that quality is primary driven by that; however, there is little empirical evidence of that.

Vinod et al. [63] try to quantify the relationship between team skills and software quality through a measuring the collective skill of the team and the quality of the software product: the degree of correlation is analyzed and inferences are made.

Evaluating collective skill

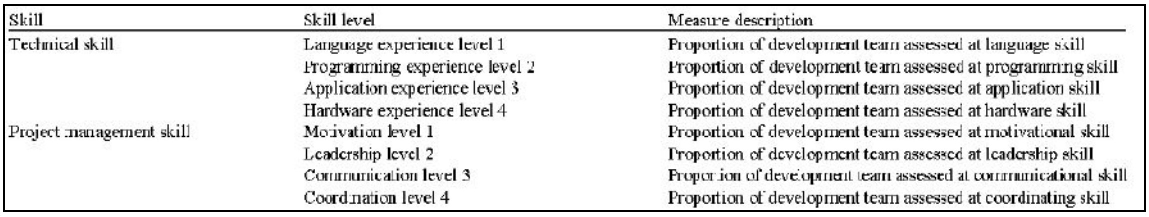
The measurement of the team's collective skills is based on the assumption that a greater skilled team will produce higher quality products. The skills analyzed belong to two main areas [63]:

- product development skills: languages, programming, hardware experience
- business development skills: project management skills, motivation, leadership, communication, coordination.

The evaluation of the collective skill of the team is based on the individual team members' skills assessments; the chosen approach for the individual assessment is NASA's CMS, competency management system, a measurement framework to quantify knowledge, skills and abilities of the workforce.

The CMS's assessment relies on a 4-tier ordinal scale which presents for each tier a set of criteria that the employee must meet in order to be assessed at that tier. Once the individual assessment is done, the whole team's skill and experience can be measured and related to life cycle phases.

The measures chosen to represent a team's skill and experience in Vinod et al. [63]'s research are presented in the table:



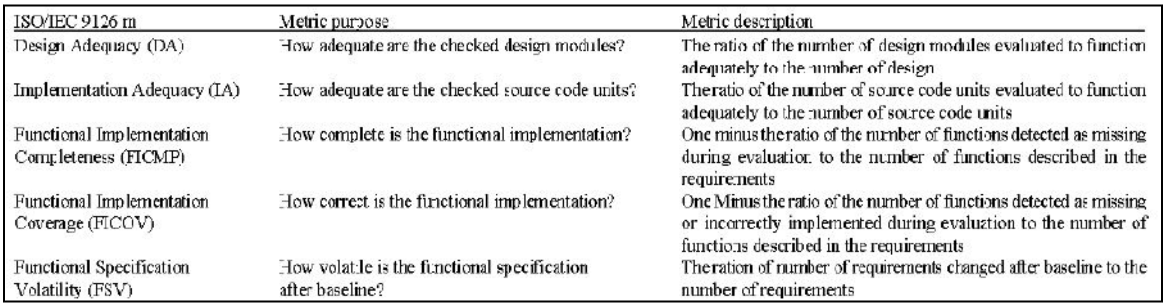
Skill	Skill level	Measure description
Technical skill	Language experience level 1	Proportion of development team assessed at language skill
	Programming experience level 2	Proportion of development team assessed at programming skill
	Application experience level 3	Proportion of development team assessed at application skill
	Hardware experience level 4	Proportion of development team assessed at hardware skill
Project management skill	Motivation level 1	Proportion of development team assessed at motivational skill
	Leadership level 2	Proportion of development team assessed at leadership skill
	Communication level 3	Proportion of development team assessed at communicational skill
	Coordination level 4	Proportion of development team assessed at coordinating skill

Tab. 4.6 Example of measures for team's skill and experience [63]

Level 1 and 2 depict marginally skilled developers, level 3 and 4 the highly skilled ones.

Deepening suitability – an example

In Vinod et al. [63]'s study, the software product quality is described referring to ISO/IEC 9126, the international standard for software quality. Six characteristics of the delivered software are covered within the standard: functionality, efficiency, reliability, usability, maintainability and portability. Vinod et al. [63] focus on suitability, a sub-characteristic of functionality that describes "the adequacy of the software product in terms of both its coverage of user needs and its implementation correctness". The metrics used to evaluate the quality in terms of suitability are presented in the table.

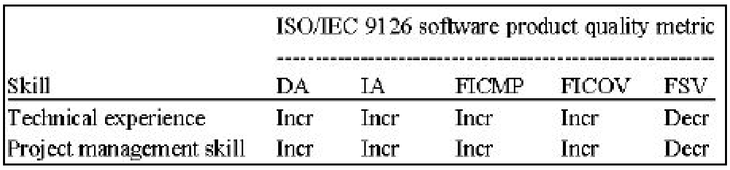


ISO/IEC 9126 m	Metric purpose	Metric description
Design Adequacy (DA)	How adequate are the checked design modules?	The ratio of the number of design modules evaluated to function adequately to the number of design
Implementation Adequacy (IA)	How adequate are the checked source code units?	The ratio of the number of source code units evaluated to function adequately to the number of source code units
Functional Implementation Completeness (FICMP)	How complete is the functional implementation?	One minus the ratio of the number of functions detected as missing during evaluation to the number of functions described in the requirements
Functional Implementation Coverage (FICOV)	How correct is the functional implementation?	One Minus the ratio of the number of functions detected as missing or incorrectly implemented during evaluation to the number of functions described in the requirements
Functional Specification Volatility (FSV)	How volatile is the functional specification after baseline?	The ration of number of requirements changed after baseline to the number of requirements

Tab. 4.7 Example of metrics used to evaluate quality in terms of suitability [63]

Skills and quality measures are recorded from 36 actual software development projects where the time frame is 3-4 months, the team size ranges from 1 to 4 developers and the development life cycle follows the waterfall model. Results are analyzed and a correlation coefficient is calculated.

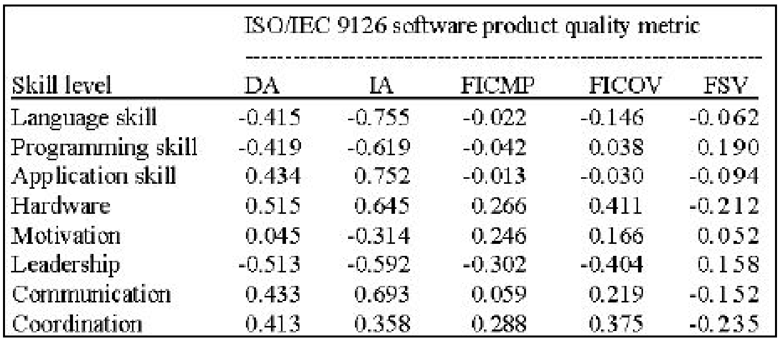
The relationship between the development teams skill levels and the software product quality metrics is described by the following table. Incr is the annotation for a direct relationship, Decr for an inverse relationship.



	ISO/IEC 9126 software product quality metric				
Skill	DA	IA	FICMP	FICOV	FSV
Technical experience	Incr	Incr	Incr	Incr	Decr
Project management skill	Incr	Incr	Incr	Incr	Decr

Tab. 4.8 Relationship between teams skill levels and software quality metrics [63]

More specifically, the relationship between each development skill (both product and business skills) and each metric is described through correlation coefficients in the following table:



	ISO/IEC 9126 software product quality metric				
Skill level	DA	IA	FICMP	FICOV	FSV
Language skill	-0.415	-0.755	-0.022	-0.146	-0.062
Programming skill	-0.419	-0.619	-0.042	0.038	0.190
Application skill	0.434	0.752	-0.013	-0.030	-0.094
Hardware	0.515	0.645	0.266	0.411	-0.212
Motivation	0.045	-0.314	0.246	0.166	0.052
Leadership	-0.513	-0.592	-0.302	-0.404	0.158
Communication	0.433	0.693	0.059	0.219	-0.152
Coordination	0.413	0.358	0.288	0.375	-0.235

Tab. 4.9 Correlation coefficients for skills and metrics [63]

Vinod et al. [63]'s study, therefore, provides empirical evidence of the effect of development team skill on the quality of the software product. However, future research is required to validate these results on a wider range of projects, for different team sizes and complexity.

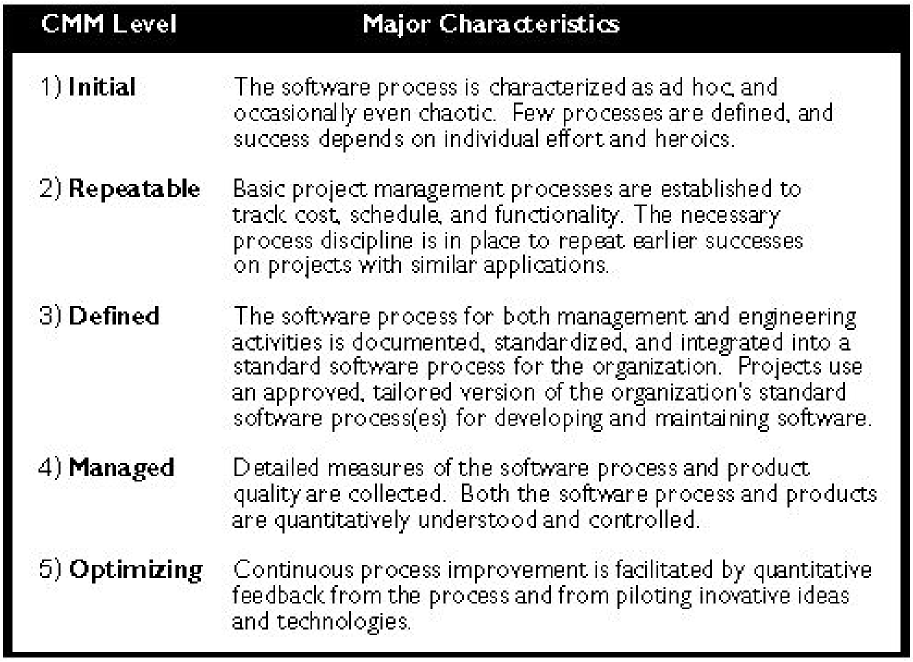
4.3.2 Software quality and the CMM

The impact of CMM on software engineering

The Capability Maturity Model intends to be a set of effective-proven practices for incremental improvements.

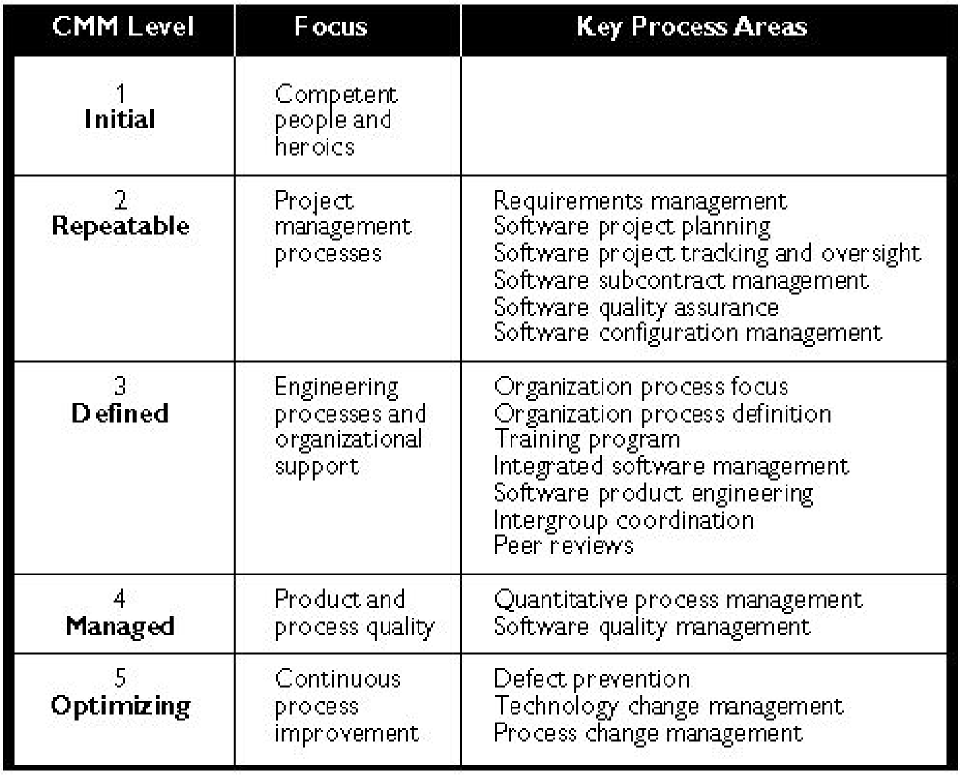
The CMM for Software (SW-CMM) aims to help software organizations progress along an evolutionary path from ad hoc to mature and disciplined software processes.

The following tables show for each CMM level the major characteristics of processes at that level and the key process areas that indicate the areas an organization should focus on to improve its software processes [25].



CMM Level	Major Characteristics
1) **Initial**	The software process is characterized as ad hoc and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
2) **Repeatable**	Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3) **Defined**	The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. Projects use an approved, tailored version of the organization's standard software process(es) for developing and maintaining software.
4) **Managed**	Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
5) **Optimizing**	Continuous process improvement is facilitated by quantitative feedback from the process and from piloting inovative ideas and technologies.

Tab. 4.10 Major characteristics of each CMM level [25]



CMM Level	Focus	Key Process Areas
1 **Initial**	Competent people and heroics	
2 **Repeatable**	Project management processes	Requirements management Software project planning Software project tracking and oversight Software subcontract management Software quality assurance Software configuration management
3 **Defined**	Engineering processes and organizational support	Organization process focus Organization process definition Training program Integrated software management Software product engineering Intergroup coordination Peer reviews
4 **Managed**	Product and process quality	Quantitative process management Software quality management
5 **Optimizing**	Continuous process improvement	Defect prevention Technology change management Process change management

Tab. 4.11 Areas of focus to improve software processes for each CMM level [25]

To study the impact of CMM on software engineering, Herbsleb and his colleagues [25] referred to the data collected by the SEI, Software Engineering Institute, through the Software Process Assessment (SPA) method. These data include the maturity level of the organization, the identified process strengths and weaknesses, the organizational scope of the assessment and the date the SPA was conducted.

In Herbsleb [25]'s study, data regarding organizations that have undergone multiple SPAs were used to answer to the following questions:

- How long does it take for an organization to move up a maturity level?

- The elapsed time between assessments in those cases where organizations moved up in level on a subsequent assessment was considered.

- What are the process challenges that distinguish those who move from the initial to the repeatable level and those who remain at the initial level?

The weakness findings were categorized according to the key process areas they served and the weaknesses were compared between those organizations that improved their maturity levels and those that did not.

Improvements

Herbsleb et al. [25] report that several case studies they analyzed show how the CMM-based software process improvement (SPI) leads to significant organizational performance to get much better. The parameters that prove this point are an increased productivity, a reduction of rework and improvements in cycle time.

However, the case studies are prone to some limitations: they may be just "lucky stories", improvements in performance may merely be a coincidence and not a consequence of the CMM-based SPI implementation or may be the result of the organization trading off other performances. Herbsleb et al. [25] undertake a survey to sort things out.

Furthermore, Herbsleb et al. [25] note that "all the reports on the benefits of process maturity come from comparisons among organizations at the initial, repeatable and defined level or from observations of organizations over time as they move through these three stages". There is relatively little experience about the higher maturity levels.

Herbsleb et al. [25] also underline the considerable amount of time and effort required for improvements to be put into place and the necessity for organizations to undergo a major shift in culture and attitudes.

Criticisms to CMM

Several criticisms have been raised to CMM: it is claimed to be counterproductive, incomplete, flawed and detrimental; it is said to lead organizations to become rigid and bureaucratic and to cause them to neglect important non-CMM issues.

In their survey Herbsleb and his colleagues [25] asked about whether any of these problems had actually occurred: from 84 to 96% of the respondents disagreed that they had experienced the problem.

Another criticism concerns organizations becoming risk-adverse for fear of losing their maturity rating. However, if risky project are no longer typically pursued, the organization is more free to add high-risk projects to its portfolio. Besides, data from the survey reports that people from higher maturity organizations see their manager as more – and not less – willing to take risks.

Three main questions

Herbsleb and his colleagues [25] identify three questions of practical concern:

1. Process maturity: how long does it take, how much does it cost and which benefits does it bring?

   Herbsleb et al. [25] estimated the median time between assessment, where organizations have moved up on a subsequent level, to be about two years.

   They also discovered that the cost and time for a CMM-based SPI program often exceeded the expectations: event if it has significant business benefits, CMM-based SPI is not cheap nor quick.

2. What are the factors that influence the CMM-based SPI success or failure?

   Herbsleb and his colleagues [25] confront organizations as they are assessed at level 1 and at the following assessment they are found having successfully achieved level 2 or still remain at level 1. The largest differences between those organizations who have moved up and those who don’t are in the areas of planning and tracking software projects, suggesting that these areas are either the most neglected or are the most difficult types of practices to put in place (or both).

   What unsuccessful organizations respond to describe their status is that the SPI “had been overcome by events and crises”; they also agree on having suffered time and resources limitations. They also claim to need more guidance about how to improve what needs to be improved.

   Herbsleb et al. [25] list those characteristics that are related to successful and unsuccessful SPI efforts:

Highly successful efforts	Less successful efforts
Senior management actively monitors SPI progress Clearly stated and understood SPI goals and responsibilities Adequate time and resources dedicated to process improvement Software engineering process group (SEPG) staffed by highly respected people Technical staff is involved in improvement Planning and tracking projects	High levels of "organizational politics" Turf guarding Cynicism from previous unsuccessful improvement experiences Belief that SPI "gets in the way of real work" Need more guidance on how to improve, not just what to improve.

Tab. 4.12 Characteristics related to successful and unsuccessful SPI efforts

In addition to the successful characteristics listed before, Herbsleb et al. [25] collect several lessons learned:

- The SPI effort requires visible support and commitment from senior management. Middle management and grassroots support and involvement is also extremely important.

- Obtaining observable results is important to keep the effort visible and to motivate and sustain interest.

- The process improvement effort must be planned, managed and given sufficient dedicated resources.

- The SPI effort must serve business interests and must be coordinated with other parts of the business in order to have foundation for the required cultural change.

3. Can the CMM be understood and applied to every software organization?

The data collected by Herbsleb and his colleagues [25] point out that there is no systematic difference in success rate due to organizational size. However they suggest that for small organizations and commercial companies CMM may be more difficult to apply.

5 Research Methods

5.1 Research design

5.1.1 Exploratory vs conclusive research

Research can be divided into two main categories: exploratory and conclusive, the latter being divided into descriptive and causal [64]. These types of research differ on their purpose: exploratory research aims to find information in order to better define a problem, descriptive research, as the name suggest, focuses on giving a better picture of an already-known problem, while causal research test hypotheses so that cause-effect relationships can be determined.

Ghauri et al. [17] relate research design, the overall plan for connecting the conceptual research problems to the pertinent empirical research [84], to the problem structure. For a structured problem a conclusive research is convenient. However, an exploratory research is the best choice when it comes to a not-so-clearly understood or structured problem, as it is "in the preliminary stages of a research project when the levels of uncertainty and of general ignorance of the subject in question are at their highest" [84]: exploratory research, in fact, is characterized by a very high level of flexibility, which "arises from a desire to learn from the experience of the investigation and from the need to avoid being blinkered by any preconceived notions" [64].

5.1.2 Qualitative vs quantitative research

Another relevant aspect when it comes to research design is whether the research should be qualitative or quantitative. A quantitative research involves some kind of measurement and the volume of the collected data should allow statistical analysis, while a qualitative research aims to identify and understand underlying motivations, attitudes and opinions, and doesn't require a high number of respondents.

The choice does not have to exclude one or the other, but what is to be investigated and how should drive the decision: "how many" and "what" indicate that a more quantitative approach should be chosen, while if "why" and "how" are involved in the aim of the research, the research itself should be essentially qualitative [84].

5.1.3 The best fit: qualitative exploratory research

The literature provides very little information on the way software companies perceive the relationship between the way they manage their people and the quality of their products.

As “the appropriate research design needs to be effective in producing the required answers within the constraints placed on the researcher” [84], for this project a qualitative, exploratory research appears to be the best fit.

5.2 Data collection

5.2.1 Qualitative vs quantitative methods

Quantitative methods

Quantitative methods for collecting data are surveys, questionnaires, experiments: they are used to collect quantitative information such as opinions, facts, attitudes, and they are usually structured and standardized. Structure reduces bias, while standardization increases generalizability, reliability, and validity.

Vidich and Shapiro [62] observe that "without the survey data, the observer could only make reasonable guesses about his area of ignorance in the effort to reduce bias."

However, often “the survey approach provides only a ‘snapshot’ of the situation at a certain point in time, yielding little information on the underlying meaning of the data. Moreover, some variables of interest to a researcher may not be measurable by this method” [15].

Qualitative methods

Qualitative methods are interviews, focus groups, ethnography: they are used to collect qualitative information such as thoughts, interpretations, meaning, and they are usually less structured and standardized than the quantitative ones.

Yin [67] suggests that case studies are appropriate if the objective of the research is to explore a previously un-researched subject.

However, as Gable [15] objects, “the conclusions drawn may be specific to the particular organizations studied and may not be generalizable”.

5.2.2 Multiple methods

An integration is possible

Integrating qualitative and quantitative research methods is possible [15] [31] [29]. Examples come from several very different fields: Gable [15] provides an example of how it’s possible to integrate case study and survey research methods in information systems, Hines [26] explains how cross-cultural research can benefit from the combination of survey research and qualitative techniques derived from cognitive science, Plano Clark [38] describes the growing acceptance of mixed methods research in social and health sciences' researches in the U.S., Borkan [6] enthusiastically pulls for their use in primary care, and so on.

How these methods are integrated: Bryman's study

Bryman [8] provides a very precise analysis of how quantitative and qualitative research are actually integrated, as his work is based on the analysis of 232 published journal articles (belonging to five different fields: sociology, social psychology, human, social and cultural geography, management and organizational behaviour, and media and cultural studies) where these methods of research are combined.

Several typologies of mixed methods are available, according to several aspects:

- Quantitative and qualitative data may be collected simultaneously or sequentially.
- Quantitative and qualitative data may have different priority.
- Why a mixed method is chosen, what function it aims to support.
- At what stage of the research process the multi-strategy research takes place: problem statement, data collection, data analysis, discussion.
- Which sources of data are available.

In his study, Bryman [8] considers articles where the multi-strategy research takes place within data collection and analysis and the employed research methods are essentially survey and qualitative interview.

The major finding is that one combination of research methods predominates in the data set: data are collected by either structured interview or questionnaire on the quantitative side along with either a semi-structured or unstructured interview on the qualitative side.

Benefits of integrated research

Kaplan and Duchon [31] say that "through the use of multiple methods the robustness of results can be increased, as findings can be validated through different kinds and sources of data".

In Bryman [8]'s study particular attention within the analysis is given to the rationales proffered for combining quantitative and qualitative research. At the beginning Bryman refers to Greene et al. [19], as they proposed five justifications for combined research:

- Triangulation: convergence of results.
- Complementarity: enhancement and clarification of the results from one method with the results from another.
- Development: use of the results from one method to help develop the other
- Initiation: discovery of contradictions or new perspectives
- Expansion: extension of the range of enquiry by using different methods.

Afterwards, Byman [8] provides a more detailed scheme that gives a better understanding of the reasons that drive the choice of integration in real cases:

- Triangulation.
- Offset: weaknesses are compensated, strengths are enhanced.
- Completeness: a deeper investigation of the area of enquiry is possible.
- Process: quantitative research gives numbers, qualitative research gives the meaning.
- Different research questions can be answered.
- Explanation: one is used to help explain findings generated by the other.
- Unexpected results of one may be better understood by employing the other.
- Instrument development: qualitative research is employed to develop questionnaire and scale items.
- Sampling: one approach is used to facilitate the sampling of respondents or cases.
- Credibility: the integrity of findings is enhanced.
- Context: qualitative research provides contextual understanding of generalizable findings of a survey.
- Illustration: qualitative data illustrate quantitative findings.
- Utility.

- Confirm and discover: qualitative data can be used to generate hypotheses and quantitative data can be used to test them.
- Diversity of views.
- Enhancement of findings.

5.2.3 The best fit: an integrated approach

Data for this project are collected through an integrated approach, where questionnaire and interview are the chosen tools. Benefits of this approach are:

- Complementarity of the methods, as weaknesses are compensated, strengths are enhanced and meaning is given to numbers
- Completeness of the investigation, as different perspective can be undertaken and different research questions can be answered.

Even though a typically quantitative method such as the survey is used, still the research remains qualitative: the purpose is to collect information to create pictures of what companies actually do, not to draw statistics out of these data.

5.3 Analysis and results

5.3.1 Analysis of the data collected through the questionnaire

Since the amount of data collected through the questionnaire is not large enough to draw up generalizable statistics (with meaningful graphics, trends, etc.), results are reported as dossiers: for each company a form, a file, is presented.

All the data collected are reported in Chapter 7.

5.3.2 Analysis of the data collected through the interviews

The interviews are recorded and the main points are noted down during the meetings.

Afterwards the interviews are entirely transcribed in order to have a detailed description of what was discussed. As the form of the interviews is more like a conversation rather than a strictly “Question + Answer” form, it has been chosen not to include the transcriptions in the thesis.

Instead, the data collected are reviewed according to the logic underlying the questionnaire (i.e. the three main themes of knowledge management, people and process management and quality management) and are presented in Chapter 7.

5.4 Closing remarks

This chapter has presented the research strategy underlying this work, which can be formalized as depicted in the following figure:

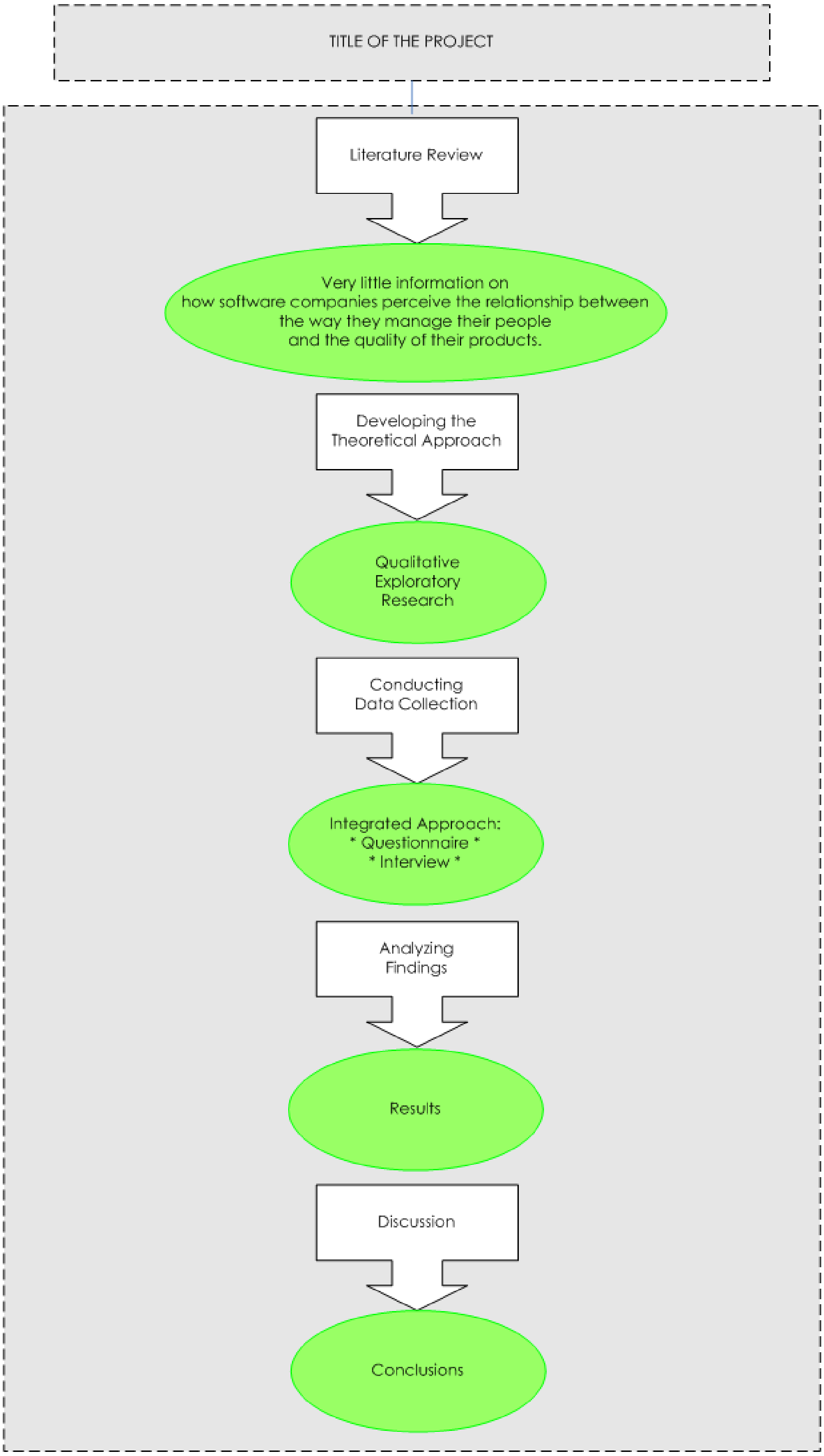

TITLE OF THE PROJECT
Literature Review
Very little information on how software companies perceive the relationship between the way they manage their people and the quality of their products.
Developing the Theoretical Approach
Qualitative Exploratory Research
Conducting Data Collection
Integrated Approach: * Questionnaire * * Interview *
Analyzing Findings
Results
Discussion
Conclusions

Fig. 5.1 Research process

6 Questionnaire and interviews

6.1 The questionnaire

6.1.1 Structure

Cover letter

The cover letter is attached to the questionnaire and aims to motivate the respondent and tells why the person was chosen, the purpose of the study, why the person should complete the survey, why the study is important, how long it will take to complete the survey, when the survey should be returned, how the respondent can see the results.

It can be found in Appendix 11.1.

The questionnaire is divided into 4 parts. Part 1, 2 and 3 mirror those main themes that have been appointed in Fig. 1.1 and aim to describe how the software company is managed and to investigate the subject of this study. Part 4 collects some general information about the respondent.

"PART 1: How do you manage knowledge?"

> In this part the respondent is asked about the importance of knowledge for his company, how knowledge is managed and shared, how knowledge sharing is encouraged and rewarded.

"PART 2: How do you manage software development and software people?"

> In this part the respondent is asked about the management, assessment and improvement of the software development process and of the software workforce.

"PART 3: How do you manage software quality?"

> Here the respondent is asked about the management, assessment and improvement of the software quality.
>
> In this part a very important section called "People management and software quality" collects several questions that look into the main theme of this study: the relationship between software quality and software people. The respondent is asked about the tools used to evaluate this relationship and the results of this kind of evaluation; strategic decisions over people

management that are relevant to software quality; and future workforce improvements and how they are supposed to enhance the quality of the product.

"PART 4: General information".

In this part the respondent's characteristics are collected: kind of company he works for, size, location, name of the company (optional) and contacts for further information (optional).

The questionnaire can be found in Appendix 11.2 (questions) and 11.3 (screenshots).

6.1.2 Types of questions

Closed questions

The questions, with the exception of those belonging to "People management and software quality", are closed questions with provided answers. However, since many questions have an "Others…" option within the answers, there are some "open" questions that are meant to give the respondent some space in order to better explain himself.

Closed questions aim to collect data for a better understanding of how software companies actually manage knowledge, processes, people and products: these questions are meant to confirm whether the well-known theories and methodologies supplied by decades of literature are actually implemented.

"People management and software quality" questions

Within the section "People management and software quality" questions are open. What is asked is very concrete and examples are requested, so that the respondent focuses on which actual things reflect his attitude towards the subject.

Opinions are not asked, since the danger is to obtain information on what everybody thinks it's right to do and nobody actually does: the use of certain tools and the capability of giving examples of certain kinds of decision are supposed to prove what goes on in reality.

6.1.3 Guidelines and tools

The cover letter has been written according to Thomas [60] and Bourque and Fielder [7]'s guidelines and suggestions.

The questionnaire has been designed following Stone [59] and Thomas [60]'s guidelines; also Fink [14] and Gillham [18]'s work were very useful.

The questionnaire has been created online as a Form in Google Docs, a web-based office suite with storage service offered by Google. This tool has been chosen because it's free, doesn't require the download of any software and it's very easy to use. Moreover, the questionnaire can be filled in online, as the respondent is given a link to the webpage, so even the respondent doesn't need anything more than a browser.

6.1.4 Benefits and limits

The designed questionnaire entails two main benefits:

- Companies located all around the world can be asked to fill in the form without the need for a meeting: lots of data can be easy collected in a short amount of time.
- As questions are mostly closed, the respondent doesn't waste time in writing and is required to put an effort only in the "People management and software quality" section, which is the most important.

Some inherent limits of the questionnaire are the following:

- Whether a closed question is not clear, there is no immediate chance for the respondent to ask for an explanation, so he may just choose a random answer and move on. Luckily, since the questionnaire is not supposed to provide statistical data, this limit is not particularly relevant.
- Whether an open question is not clear, there is no chance for an immediate dialogue, so the respondent may write something that has nothing to do with the question or just skip it. Also, some answers may be worth being deepened, but without a further contact it's impossible to do. This is a major limit of this choice.
- Companies may be unwilling to answer to an online survey without knowing who made it and what will be done with their answers. To make up for this, a letter explaining the purpose and the context of the study introduces the questionnaire.

- “Essentially, the researcher should have a very good idea of the answer before starting a survey. Thus, traditional survey research usually serves as a methodology of verification rather than discovery” [15]. In this case the survey is used as a tool for discovery, since the research is exploratory.

6.2 The interviews

6.2.1 Structure

Each interview should take about an hour.

The intent is to collect information about what happens in a real situation, therefore the interview should aim to allow the interviewer to form a picture of how the interviewed company works.

Moreover, the interview should cover the three main topics addressed in the questionnaire: knowledge management, people and processes management and quality management. The interview should aim to position the company in respect to each of these topics, defining what’s the overall culture and what’s the actual situation.

The questionnaire should be used as a point of reference, in order to cover all the main aspects that are relevant to form a coherent picture.

6.2.2 Guidelines and tools

The interviews have been conducted according to Canavor and Meirowitz [9] and Gubrium and Holstein [20]’s guidelines and suggestions.

The interviews have been recorded and transcribed later on. During the interviews the main points have been noted down in order to keep track of the topics already discussed and of those that needed further elaboration.

Photos about the Scrum room and the white board used by the companies have also been taken (see Appendix 11.4.4).

6.2.3 Benefits and limits

An interview presents the following benefits:

- A face-to-face meeting can prove that the interviewer can be trusted.
- Questions and answers can be accurately explained.
- Unexpected aspects can emerge.

An interview has a major limit:

- It takes time, as people tend to wander while talking.

In this particular case, the different mother tongue and culture (Italian vs Swedish) may also add some difficulties, so particular attention should be given over to ensuring a full understanding of questions and answers.

7 Results

7.1 Data from the questionnaire

7.1.1 Form n°1:

It’s the IT department of a service company whose size is small (less than 50 people) in Italy.

The intellectual capital of the company is estimated to be responsible for more than 80% of the total added value of the products.

Knowledge sharing is explicitly encouraged, even though there is no explicit strategy for that: people are free to share knowledge as they prefer. No particular tool is used: common tools such as telephone, e-mail and meetings are used as the need of knowledge sharing arises.

There is no explicit methodology neither for knowledge management nor for process management. No methodology is applied for managing, assessing and improving the workforce either.

There is no systematic approach to software quality and quality is not believed to depend on how people are managed.

7.1.2 Form n°2

The type of company is not specified, but it’s a medium-size company (between 50 and 250 people) in Italy.

The intellectual capital of the company is estimated to be responsible for a total added value of the products that ranges from 65% to 80%.

Knowledge sharing is not considered a managerial issue and there is no explicit strategy for that: people are free to share knowledge as they prefer. No particular tool is used, but common tools such as telephone, e-mail and meetings are used in a standardized way, according to the particular need..

There is no explicit methodology neither for knowledge management nor for process management. No methodology is applied for managing, assessing and improving the workforce either.

There is no systematic approach to software quality. However, quality is believed to be affected by the way people are managed.

7.1.3 Form n°3

It’s a medium-size software house (between 50 and 250 people) in a country that is not Italy nor Sweden.

The intellectual capital of the company is estimated to be responsible for a total added value of the products that ranges from 50% to 65%.

Knowledge sharing is explicitly encouraged, even though there is no explicit strategy for that: people are free to share knowledge as they prefer. There is a dedicated IT system for knowledge sharing that is made up of several tools: knowledge tree, Microsoft SharePoint and internal wikis. However, no explicit methodology is used for knowledge management.

The software development process is managed, assessed and improved according to CMMI (Capability Maturity Model Integration) methodology, while the management, assessment and improvement of the software workforce doesn’t rely on any methodology.

Quality is approached according to customer satisfaction. Quality is believed to depend on the way people are managed and strategic decisions have been made over people management to affect the quality of the product: the example given is about dedicated focus / investment in QA (quality assurance) teams. Moreover, in order to improve the level of skills and competencies of the company’s software people, internal training are planned.

7.1.4 Form n°4

It’s a small European software house (less than 50 people).

The intellectual capital of the company is estimated to be responsible for more than 80% of the total added value of the products.

A people-to-document approach is used for knowledge management: knowledge is extracted and codified from the person who developed it, so that any other person can search for it without having to contact the original owner. No specific tools are used, but people use common tools (such as telephone, e-mail, meetings) in a standardized way, depending on the particular need.

Knowledge sharing is explicitly encouraged, but not rewarded. Also, there is no explicit methodology applied for managing knowledge.

No methodology is applied neither for the management of the software development process nor for the management of the workforce.

Quality is approached according to customer satisfaction, as customer requests are listened to and are systematically taken into account when developing the software. Quality is believed to depend on the way people are managed.

7.2 Data from the interviews

7.2.1 Interview n°1

The manager interviewed was Magnus Werner, Acting Director for Windows Software at SAAB Training and Simulation.

Software development process: agile development with Scrum

It's SAAB philosophy to create independent, self-managing teams that are fed by the managerial level with tasks, the "what to do", and are left free to do the planning and come up with solutions to the "what": the "how to do it" is a team's responsibility to define.

The idea beneath the management of the software development process is that software developers want to start coding as soon as possible. In consideration of this, SAAB has chosen not to follow the traditional waterfall model (i.e. requirements specification and analysis, software design, integration, testing, installation, maintenance), but to refer to Agile development instead. The main advantage of Agile methodologies is that the gap between the specified requirements (what should be produced) and the final product (what is actually produced) is considerably reduced, as iterations are undertaken and customers and stakeholders are much more involved.

SAAB applies Scrum, an iterative, incremental process with roles and practices. It's really simple and its description can fit in a poster, that is *"…pretty much how much a developer can take"*.

There are three main roles:

- the Product Owner, who is responsible for the long-term perspective and is the interface with the customer, as he has the knowledge on customer needs and features required.

- the Scrum Master, the team leader and project manager, who acts as a coach and facilitator for the team, and whose role depends on the maturity of the team itself (i.e. for how long the team has worked with the Scrum framework).

- the Scum Team, the group of developers that will produce the software. The maturity of the team can be assessed looking at the number of points committed and the number of actually completed (if there is a big difference it means that the team has overestimated or underestimated its capability), even though the context of the stories (the features to implement) should be considered: if new functionalities are to be developed or if new technologies are used it's possible that even a mature team cannot be able to complete all the stories it has committed to do.

All the requirements, bugs or issues that have been found as well as new ideas are coded into "stories" that made up the so-called Product Backlog: a collection of cards where these stories are described. The Product Owner prioritizes the stories in the Product Backlog – two stories cannot have the same priority, there is always something more important and something less.

An iteration, called Sprint, lasts three weeks. At the beginning of each Sprint, a Sprint Planning Meeting takes place: during this meeting, attended by the Product Owner, the team and customer representatives and other stakeholders, the work to be done is defined through the Product Backlog. The team gets to estimate the difficulty of implementing each story in the Product Backlog: this esteem is a measure of the capability of the team. The quantity of stories to be done within a Sprint is a negotiation between the Product Owner and the Scrum Team. Also a document about the time and resources required, called Sprint Backlog, is written.

SAAB keeps statistics about the maturity of the teams, the number of stories to whom the teams have committed, etc. Statistics are drawn up not only within the software development department, but all over the company. At the moment these statistics are obtained through the use of Microsoft Excel and are showed through diagrams and graphics that can be seen in the corridors near the managers' offices. However, SAAB is planning to introduce a new system, a Microsoft product: Team Foundation Server, an application for lifecycle management (ALM). It is supposed to help in the process of collecting data and in the chain management, it *"…tries to please everyone"* (e.g. developers by providing what they need, like way to check out source code, or planning when to introduce new functionalities and features or to fix bugs and solve issues; managers by providing tools for doing several types of analysis). This will absolutely not substitute the white board that nowadays supports the Product Backlog cards and planning (that can be seen in Fig. 11.5, Appendix 11.4.4): the board and the pieces of paper with the stories and the coloured post-it, could be put into the tool, but this would take lots of time. Moreover, if there isn't a big monitor where everything can be displayed (as it is now with the board) then the visualization disappears, and the immediate access to the information on how everything is going – that now can be achieved just by taking a walk to the Scrum room – is not there anymore.

Why is visualization so important? Because *"…when you have everything just in front of you, you're always finishing on time"*. SAAB wants to promote visualization and this is one of the reasons they work with Scrum, show diagrams on the walls of the management level and much more.

Everyday, at the same time and place (in the morning, within the Scrum room), a meeting called Daily Sprint takes place. Three questions are asked to the team: “What did you do yesterday?”, “What will you do today?” and “What’s in you way” (i.e. “Did you have any problem?”). The team updates the Sprint Backlog and the Burn Down Chart, which is the Scrum Master’s *“baby”* – as it says whether the Team is on time or not. The Product Owner also attends each Daily Scrum: he may not speak, but the team may have questions about the stories or may think that it won’t be able to meet the commitment and asks to remove a story or two (it’s a call on the Product Owner to decide which stories to remove).

At the end of each Sprint, a Sprint Review Meeting and a Sprint Retrospective are held: during these meetings what has been done is presented and how the Sprint that is just finished has gone is discussed: high points, low points, improvements to make the next Sprint even better – thus, for each Sprint there is a list of things to try to improve. For example, each story has a counting number, that is the time to spend working on it, so the Scrum Master has to learn to correctly assess these hours. Another example is the decision whether to increase the level of testing during each sprint. Improvements are focused on the process.

More about Scrum can be found in Appendix 11.4.4.

Knowledge management and sharing

“I don’t believe in systems, you’ve got to know people. As a manager you have to know the strengths and weaknesses, you need to have a good picture of what they know and – even more important – their personality and what they like to do”.

At SAAB there is no dedicated IT system for knowledge sharing, as the philosophy is to create the right mix of people for a team and putting them together in a big room. The purpose it to promote communication in a simple and spontaneous way, so if someone needs help he can just shout out “Does anyone know how to fix this?” and hopefully someone within the team does. If not, it’s up to the individual to search for help though the network of people that is naturally constituted within the company. The company expects its employees to be as autonomous as possible: given a problem, try to figure out on your own, ask the room, the team, search the Internet, etc. – if you get really stuck, then go to the next level, ask a manager for help.

The network is enhanced through making people work together in projects and often cover other roles than their own: if you are a developer you are expected to be able to help the person who’s testing or write a document if you are asked, for example. *“It’s really important that people understand how they fit* *in the big picture – and to make them really aware of that the best way is to force them to do other people's work"*.

Moreover, it's important to have people with competences. At SAAB the level of education is quite high: lots of engineers and PhDs are employed. Also, people are encouraged to take courses to expand and deepen their area of expertise (the company is available to pay for courses, books and certifications), even though it's up to the individual to decide which courses he wants to attend. When a new system is introduced, courses are held within the company too.

Twice a year, in autumn and in spring, a "utvecklingssamtal" is held with each employee: it's a meeting where the individual's future knowledge enhancements are discussed. The employee is asked about what he would like to do and a plan is drawn up (e.g. he should take this course, he should read that book). Most people tend to focus on enhancing their knowledge rather than their interpersonal skills, but skills on how to work together are taken into account – for example recently a half-a-day seminar on how to give feedback in order to improve team work has been held.

Knowledge sharing is explicitly encouraged also through the philosophy that what matters is the team's work, not the individual one. Everyone's priority, as a team member, is to contribute as much as possible to make the team successful.

This organizational culture, this particular mind set, together with the shared office, is what prevents the people from being lazy: it's really hard not to work, because the person next to you will notice.

Software quality

The main indicator of quality is the number of bugs reported by the customer each month. In order to reduce this number, problems and errors must be discovered and solved before the software is delivered to the customer.

The Daily Scrum is what helps in constant monitoring, approaching problems and assigning the task of solving them.

Regression testing, that is to make sure that the new code hasn't damaged the old one, is something on which SAAB has invested a lot. In fact, as a system grows it's really important to keep an eye on the interactions between the old and the new functionalities. That's one of the reasons that led SAAB to the decision to use Team Foundation Server, as the Microsoft tool provides a way to make a faster regression testing (the alternative is to hire more people just to have them *"sit and press a button"* to check that everything still works).

At SAAB there is no standard approach when it comes to quality management. It's something on which it may be worth reasoning, but it hasn't been planned yet. However, it's worth notice that the high customer involvement due to the use of Scrum (i.e. the presence of the Product Owner to every daily meeting to remind the customer requirements and needs) has led to a very customer-oriented approach to the whole software development process, quality included.

When it comes to quality, the way tests are performed is essential: *"What we have seen is that we have had several cases where we tested systems in the house and it seems to work quite ok and then, once it has been delivered to the customer and it's in production-mode, issues surface because it's used in a slightly different way from how it has been tested in the house"* (e.g. inputs are given in a different order). So, customers sometimes report problems that the team may even not be able to reproduce, least of all to forestall. Since testing a system with data from a real situation is so relevant in order to deliver a better product (because errors have been found and solved in the house and the number of bugs found by the customer is therefore reduced), the team is always looking for a tool to record and report data from the production-mode, the customer's "environment".

People management and software quality

"The most important thing for good quality software is to have good quality employees. You really need to have competent people working for you, because if you don't you'll be in big troubles […] And you need to have people that actually are happy to go to work, keep people happy and pleased so that they enjoy their work. If they have fun at work they usually produce good stuff".

To have quality people and a quality environment, which are believed to be decisive for software quality, several strategic decisions have been made:

- Hiring competent people with a high level of education: people and their knowledge play a basic role in the realization of such particular product as software.

- Adopting Scrum, which is iterative and incremental: it allows developers to start coding very early in the process, that is what they want the most.

- Pursuing a culture of knowledge sharing, encouraging autonomous problem-solving, locating all the team members in one room: it acknowledges each person's competences and provides motivation.

- Making people aware that what matters is the team work and having them capable of covering other roles than their own: a reduction of the lead time (i.e. time required to deliver the product) can be considered an improvement in quality.

- Sustaining, promoting and enhancing the development of a network of people through making people work together in projects and having the time to meet and discuss: it allows knowledge sharing in a simple and “popular way” (as people rather talk to each other than look into a database).

There is no explicit methodology to manage the relationship between people and software quality: it’s simply assumed that better people and a better environment lead to a better quality and decisions for achieving the formers are made.

However, once a year a survey about employee satisfaction and “happiness” is undertaken by the employees. Whether a decrease in the level of overall well-being in a certain department is noticed, it’s further investigated – even though often there is a natural explanation such as an organizational change.

Additionally, the level of well-being is constantly monitored by the managers. There is no dedicated tool for this: *“…the big part is a feeling […]I think it’s equally important for managers to use tools such as a survey and the chit-chat or just walking around and observing”.*

The basic idea is that a manager should speak with his people, get to know them, work actively with them, so that he knows their capabilities, personalities and feelings about what they are doing. Since most people are reluctant to changes, but on different levels, part of the manager’s job is to be able to handle that too.

“It’s much about finding the right place for each person, putting people in a comfortable context where they can perform well”.

7.2.2 Interview n°2

The manager interviewed was Anna Leo, product manager at the IT Department of the Swedish Board of Agriculture (Jordbruksverket).

Software development process: agile development

Methodologies used in the management of the software development process are Agile, Scrum, and RUP: the development process is built upon RUP and relies on Agile principles, and the skeleton process provided by Scrum is also used. These many principles and procedures may appear to be too many, but they are not used all together, even though a pilot process is about to experiment their combined use.

Right now, they are used according to what has to be done, in order to have something to hold on to, something to follow, that gives indications on what should be done to achieve greater efficiency. These methodologies have proved to be helpful in order to reduce the time required to solve problems – which leads to an increase in quality (where quality is intended as the lead time required to develop the product).

The main principles of Agile development are applied: Individual and interactions, Working Software, Customer collaboration, and Responding to change. This leads to the fact that the three main characteristics of Agile development – timeboxed iterations, teamwork, and customer involvement – can be tracked as well. The following table presents the Principles of the Agile Manifesto in details as well as their implementation at the Swedish Board of Agriculture:

Principles of the Agile Manifesto	**How these principles are implemented**
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	The choice of using Scrum assures customer satisfaction as what the customer wants is reminded every day to the Scrum Team by the Product Owner. Moreover, organizing the development according to the Scrum skeleton process allows to have software delivered at the end of each Sprint, that is two weeks. Also, one of the six RUP's best practices is "Manage requirements", that is keep in mind what the customer wants. Moreover, Agile Modeling addresses "Prioritized Requirements": requirements are implemented by the agile team in priority order to provide the greatest return

	on investment (ROI) possible.
Welcome changing in requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Scrum makes really easy to embrace changing in requirements, both because requirements are not fully defined in advance and because at the beginning of each Sprint what appears to be important to the customer is reviewed (so what's new can be introduced within two weeks). Also RUP addresses the fact that requirements are not needed to be all defined in advance, as one of RUP's best practices says "Develop iteratively, with risk as the primary iteration driver".
Deliver working software frequently, from a couple of weeks to a couple of months, within a preference to the shorter timescale.	The use of two-weeks Sprints leads to the delivery of working software frequently.
Business people and developers must work together daily through the project.	A Daily Scrum is held every day and attended by both developers (i.e. Scrum Team) and the Product Owner, who represents the customer and other stakeholders. Agile Modeling enhances "Active Stakeholder Participation": the stakeholders should provide information and make decisions in a timely manner and be actively involved in the development process.
Build project around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	In order to have people working according to Agile, Scrum and RUP two things are important: to have people developing the methods so that they fit better the situation, and to have people who teach to others how the methods work (both when a new method is at first introduced and within the projects).
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Daily Scrum is held to discuss what has been done, what has to be done, and the problems emerged. Working in open offices is what enables information sharing during the rest of the day.
Working software is the primary measure of progress.	Tests are done very often, so what is considered "Done" is working for sure. What is done is represented through the Scrum's Burn Down Chart, so it's constantly monitored.
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Scrum's Sprints (plus the Daily meeting) assure that a constant pace is maintained.

Continuous attention to technical excellence and good design enhances agility.	One of RUP's practices tells to "Employ a component-based architecture", that is to break down complex projects into smaller parts to favour code reuse. Another practices says: "Continuously verify quality through tests".
Simplicity – the art of maximizing the amount of work not done – is essential.	Agile Modeling summarizes this concept as "JBGE – Just Barely Good Enough": a model or document should be sufficient for the situation at hand and no more. Also Scrum assures that what is done is what the customer values.
The best architectures, requirements, and designs emerge from self-organizing teams.	Scrum, RUP and Agile Modeling provide a set of practices and principles: the team is free to move within them in the best way to achieve a good quality product.
At regular intervals the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.	Scrum's Sprint Retrospective is exactly about that, as what was good and what was bad within the Sprint is asked, in order for the team to be more effective in the next Sprint. RUP says: "Control changes", which means keep an eye on what affects the team's efficiency (team structure, location, platform choices, …).

Tab. 7.1 Agile principles and their implementation at the Swedish Board of Agriculture

More about Agile methodologies (Agile Modeling, Scrum, RUP, …) can be found in Appendix 11.4.4.

Knowledge management and sharing

When a new project is about to begin, a project leader is selected: he or she looks into what should be done and defines what kinds of people are needed to do the job. The managers responsible for the workforce are asked by the project leader about people to cover the roles and it's their responsibility to find the right people for those roles, even though the project leader may suggest some names.

People to cover the roles within the project are found within the company or outside: the company employs about 65 people and 40 – 50 consultants are available too. Note: customers are also involved in the project; customers can be internal or external (in which case they may send a representative).

The main capabilities and knowledge and the role of each person are recorded in an Excel document that is filled in by the managers responsible for the workforce according to the information gained through the initial interview and updated during the way.

While the project is being carried on capabilities and knowledge that cannot be found within the team may be required: in this case the project leader asks for managers of the upper level to take care of this. The managers responsible for the workforce can look for what is needed both in the house of out in the city. Thus, there is hierarchical structure: when there is a problem, it progresses to the upper level.

However, most of the time problems can be solved within the team – during the Daily Sprint help is required, there is an unvoiced assessment of each team member’s capabilities, and the team decides who should help.

Most of the employees at the Swedish Board of Agriculture have a software development degree. The employees can upgrade their knowledge and capabilities through taking courses, attending meetings, taking certifications, etc. Usually when they find something interesting they ask their manager whether they can do it or not and it’s the manager’s call on that. As there is always much going on in terms of methodologies and tools, people attend courses quite often. Also courses about how to work together are taken. All these courses are mostly short ones.

It should not go unnoticed that people may work at more than one project at the time: even though this may lead to a lack of focus, it’s still accountable as a mean for knowledge to spread.

Software quality

A philosophy underlies the management of software quality: doing the right thing, that is *“… decide what is the most important thing to do and do it first”*. What is important is what matters to the customer.

Quality is evaluated in terms of number of errors and it is checked the whole time through tests: for each project there is a test leader, a person who tells the product leader how much has been tested and how many errors were encountered.

Quality is also seen as customer satisfaction: in order to assess it, customers (intended as final users) test the product the whole time too – “*…the customers are IN the project*”. Meetings with the customers are both scheduled and held when an error is found.

The interaction within the customer and the software is tested too, through what is called usability testing, in order to achieve a better understanding of what the system is used for (i.e. what the customer actually wants).

People management and software quality

Even though it’s believed that there is a relationship between software people and software quality, there isn’t a specific tool, statistics or methodology to assess it – managers prefer talking and having face-to-face contacts with their people.

“*The quality of the product depends on whether the people that develop are feeling good. I think it’s very important to talk with the people that are in the project and ask directly to them what’s good and what’s not*”.

Apart from everyday contacts, there is a scheduled moment for people to share their feelings about their jobs and about the project: the Scrum’s Sprint Retrospective. During this meeting the team is asked about what went well (“What should we keep doing?”), what went bad (“What should we stop doing?”), and what can be done better (“What should we start doing?”). Therefore, strategic decisions are made according to the answers to these three really simple questions.

8 Discussion

8.1 Software and intellectual capital

8.1.1 Knowledge and competences: inputs and tools for the production of software

People in software companies estimate the intellectual capital to be responsible for a percentage of the total added value of the products that ranges from 50% up to more than 80%, which reflects what is said by Schreiber et al. (75% to 80% of the total balance sheet) [68], meaning that they recognize the leading role of human competences in the process of software development.

Knowledge and competences are required both as inputs and as tools for the realization of the software product: in fact, knowledge and competences are cut in on the development process as well as in the management of people, processes and quality.

8.2 Knowledge management

Knowledge management, that is the management of the knowledge that is used as an input for the realization of the product, involves two main activities: knowledge sharing and knowledge management (i.e. the management of knowledge as a resource) and enhancement.

8.2.1 Knowledge sharing

Person-to-person approach: a network of people that should be mapped and more effectively supported

Knowledge sharing is approached in several different ways, but mostly it relies on the natural formation of a network of people within the company that communicate through common tools such as telephone, e-mail and formal and informal meetings. Even though they are not rewarded for that, people are nudged to share – moreover, as the interview with SAAB's manager Magnus Werner revealed, the company expects its employee to share.

It may surprise that software companies still rely on the old-fashion person-to-person approach, but when dealing with a customized product (that is the case of software for the interviewed companies), the choice of a personalization strategy is the more appropriate one. The use of common tools for communication instead of relying on more innovative and technological devices falls within the explicit advice of Hansen, Nohria and Tierney [22]: "Do not straddle", follow one strategy at 80% and the other at 20%, do not try to have both the personalization approach and the codification approach implemented. In fact, as the IT support required should reflect the chosen strategy, for a personalization strategy a system that allows people to find each other is what should be provided.

A common choice to foster knowledge sharing is to have people from the same development team to work within the same room, usually an open office. Also, as hinted before, an organizational culture that recognizes and celebrates experience sharing is what allows the sharing to be concrete. Methodologies such as Scrum that involve daily meetings provide further points of contact for people to establish relationships.

However, even though the chosen strategy is to have knowledge shared through person-to-person contacts with the support of common tools, it appears that these contacts could be more effective. Managers are prone to say that people know each other, as they work together, so "who knows what" is of common understanding. Still, even though technology should not aim to replace the existing sharing processes, the sharing could be more effective if there was a system to classify “who knows what” (e.g. some kind of Yellow Pages) and if that system was available to everyone. An Excel file, as it is at the Swedish Board of Agriculture, could be a beginning – but it should be accessible to all the employees, as now it’s not, and it should be updated on a regular basis. The drafting of a knowledge map as well as the identification of the human roles within the so-called knowledge community, as suggested by Junnarkar’s methodology [30], could be further steps.

Codification strategy is chosen for standardized products and knowledge engineering should be considered

On the other end, when it comes to standardized products, a codification strategy is more likely to be chosen. For a computer-based approach something like a library, containing documents and allowing searches, is needed for knowledge sharing.

This would mean both providing an up-to-date mean for collecting and sharing information and experiences and providing people with incentives for sharing, as the “social reason” goes missing (people-to-people strategy involves sharing as an activity for being part of a network, while people-to-computer strategy needs to reward people for putting information within the system).

In this case there would be no need for a map of “who knows what”, as everyone “knows” (or, at least, can found information about) everything. However, a close look to what is called knowledge engineering should be taken, as it provides a wide range of methods and techniques for the acquisition, modelling, representation and use of knowledge.

8.2.2 Knowledge management and enhancement

Knowledge management is not explicitly mentioned: a strategic perspective is missing

It’s worth reminding that knowledge as a resource has several distinctive characteristics: it’s intangible, difficult to measure, embodied in an agent, not consumed in the process and on the opposite may increase through use, cannot be bought on the market, and has a wide impact on the organization.

Knowledge management is not explicitly enlisted as a managerial issue in software companies, as concepts such as the knowledge-management cycle and the knowledge value chain are not mentioned by the interviewed and surveyed managers.

Nonetheless, some of the basics underlying these concepts can be spotted in the way software companies operate within the area of knowledge sharing: for example, having people working together in teams and having them able to cover several other roles rather than their own (as it happens as SAAB) leads to the awareness on “who knows what” as well as to knowledge spread. Moreover, the “Reflect” concept that belongs to the knowledge-management cycle can be partially spotted in those companies where the organizational culture suggests and supports the individual choices over attending courses, studying, obtaining certifications, etc., and even more where an actual plan concerning the individual’s growth in terms of competences and skills is done and reviewed (at least) every year.

Still, a main limit can be pointed out: knowledge management and competences growth are handled only on the individual level and there is no organizational strategy for monitoring the current situation (such as having a record at the managerial level on “who knows what”, “who needs what”) nor for planning and implementing actions (which knowledge should be built over time and how).

A strategic perspective over knowledge management appears to be missing. Given the great importance of knowledge, as it is the main input when it comes to the production of software, software companies should more carefully consider putting to use an explicit approach to the management of this resource. The knowledge-management cycle can provide general directions in this sense; advices and concrete guidelines and tools are Wiig’s tables [66], Wielinga et al.’s onthologies [65], as well as the whole Junnarkar’s methodology [30].

Moreover, as for manufacturing products Porter’s value chain is very frequently mentioned, for such products as software – that rely consistently on the intellectual capital – it’s worth considering the knowledge value chain [46]. Identifying the knowledge value chain allows to formulate more easily a strategy for the management, assessment, and improvement of knowledge.

8.3 People and processes management

8.3.1 Software development process

Methodologies are used, but not systematically assessed, nor are assessed the single individual practices

Activities such as software specification, design, implementation, validation and evolution are organized differently according to the several different models: waterfall model, evolutionary development, component-based software engineering. In reality, these models are often combined, giving rise to a wide range of methodologies.

Nowadays software companies tend to be oriented towards those methodologies that take into greater account the need of being ready for change, mainly for what concerns customer requirements. In particular, great success has been achieved by Agile methodologies such as Scrum, RUP, XP, etc., as those methodologies are characterized by iterations, continuous improvement and high customer involvement in the process, as well as the possibility for developers to start coding much sooner (which is all they ask for). Above all, these methodologies are used because they provide something to hold on to, they give indications on what should be done to achieve greater efficiency.

However, the assessment on whether the implementation of these methodologies brings actual benefits is not scientifically done. For example there may be daily meetings and meetings held at the end of each iteration (as it is with Scrum) where “what is good”, “what is bad” and “what else should be done” is discussed, but the use of Scrum itself is not evaluated at a managerial level.

What appears to be missing is the actual managerial assessment on whether the chosen practices and the implemented methodologies are bringing benefits and on how processes are improving. The only exception detected is a medium-size software house that applies CMMI (Capability Maturity Model Integration).

Frameworks such as CMMI or SPICE should be used because they provide guidance on defining objectives, planning and executing improvements. Therefore, they help managers to achieve a better understanding of which choices in terms of process management lead to a greater efficiency – and this understanding is very concrete, as it is based on the observation and monitoring of their own reality.

The single practices carried out by the workforce when it comes to software development should be analyzed too, as personal process discipline can lead to individual effectiveness (and consequently to improvements in the performance of the team the individual belongs to). Software developers should be motivated to find and adopt effective methods, but very often this is a task that is left to the individual's own considerations, as sound methods are not provided.

Instead, software people should be trained for planning and evaluating the methods they use. It may appear a task for managers, but having it carried out by developers has a valuable benefit: it leads to a better understanding of one's own performance and motivates the search for further improvements and new best practices. SEI's Personal Software Process (PSP) is an example of a structured, disciplined and measured process that provides guidance in analyzing and assessing the developer's practices and deciding of further improvements.

8.3.2 Software workforce

People management is not a task for HR only

The people working in a software organisation are its greatest assets: a poor management of people is one of the most significant causes of project failure. People should be chosen carefully, motivated and rewarded. Software development is very often carried on by teams, so the way a team is made up is a critical task too. That's why the way people work and how they work together should be analyzed.

This work doesn't aim to provide sociological advices, but it is far beyond the pure HR's interest to investigate organizational attitudes, social networks, and team interactions (e.g. stages of group development, roles and status, leader and followers, etc.): it is also in the interests of managers to have a better, deeper and more scientific understanding of the people they manage.

Workforce maturity should be assessed: a methodology should be used

This work focuses on the assessment of the workforce maturity, that is how disciplined the development of knowledge, skills and motivation of the workforce is.

A beginning is what is done with Scrum: the statistics about the number of stories to whom a team has committed tell the manager something about the maturity of the team, how "good" the team is (how efficient, how well-organized, etc.). However, those companies that have been surveyed and interviewed don't report to have an explicit methodology for dealing with maturity assessment. It’s a judgement left to managers, based on their impressions.

The lack of a methodology for maturity assessment is something dangerous, because it means that there is no tracking of how teams evolve in time, if they make progresses or if they get worse. In fact, the problem is the absence of a strategic, long-term perspective: a manager may know how a certain team is performing, may have an idea of the relevant performance improvements in the past, but he may as well not be able to see the trend, the slightly worsening that take place day after day, and definitely has no planning over where he wants the team to be from here to the next year (or two, or five years…).

P-CMM is a framework for assessing and improving the way the organization manages its human assets: this is particularly necessary when it comes to the software-development industry, as it is characterized by talent shortage and constant change in technologies and programming languages and therefore requires that software companies not only recruit, train and retain their workforce, but also keep continuously improving the workforce’s capabilities. P-CMM is suggested here as it is reckoned to be the most comprehensive framework for recognizing, standardizing and improving the use of good practices for increasing the workforce capability and expanding this capability beyond individuals to the whole organization. P-CMM has the great strength of being made up of practices that have been proved to be effective and, above all, thanks to its structure (the five levels) it provides a path, an actual roadmap for organizations to plan and carry out their growth in terms of workforce maturity over time.

8.4 Quality management

8.4.1 Define quality

Quality for the interviewed software companies means mainly two things: number of errors detected and customer (user) satisfaction. There is no explicit strategy: it's just assumed that the software has to work and it has to do what the customer wants it to do.

Constant monitoring through Scrum meetings, regression testing, usability testing and customer involvement in the process are the tools that aim to assure that quality is achieved.

A lack of a strategic vision is once more detected. Software companies seem to be focused on the "right now", they appear to have the goal of making the software work and the customer happy, but they don't have a strategic vision over how they achieve this goal.

8.4.2 Measure quality

Quality is hard to define and it's even more hard to measure. However, much more than simply listening to the customer, trying to make a good software and testing it for errors can be done: a wider collection of factors that are relevant for quality (e.g. understandability, completeness, reliability, portability, etc.) should be considered, and data should be collected and analyzed to have a better-defined picture of the level of quality achieved.

Moreover, an explicit approach to quality should be defined. What is important to who and why should be of common understanding in order to focus the developers' efforts. Also, guidelines on how to achieve the desired quality should be provided.

Standards such ISO 9000 define quality as the conformity to requirements specification and provide guidance on how to develop quality plans to support quality management as well as general quality principles and specific procedures to define; the international standard ISO/IEC 9126 described quality as a set of characteristics of the product (functionality, reliability, usability, etc.) and breaks them into measurable product attributes.

Approaching quality with the focus on customer satisfaction can be done as well, as authors like Denning [13] depict what customer satisfaction actually is and give directions about the stages to be undertaken to achieve it.

Garvin [16] and Bevan [4] consider quality from the user perspective: effectiveness, efficiency and satisfaction are mentioned, and user centered design is suggested. In order to evaluate quality from this perspective, the MUSiC methods can be used.

The quality of the product is also believed to be affected by the quality of the process, therefore managers can choose to undertake the point of view of the development process and consider concepts such as quality of design and quality of conformance and assess software characteristics such as reliability according to the different stages of the process.

8.5 People and quality

8.5.1 More attention should be given over people and quality management

A structured approach to processes management is well-established, as nowadays managers are used to think in terms of processes and operations.

Rising attention is due to people. Swedish companies in particular are committed to providing a good environment for people to work in, as they recognize that those employees that “feel good” perform better. Daily meetings to discuss ideas and problems, half-yearly “utvecklingssamtal” to analyze and plan the worker’s competences growth, and yearly surveys on whether the employees are happy about their job are, at the moment, the tools that companies use to assess workforce satisfaction.

Quality is being discussed too, as software companies are always striving to deliver a better product, with less errors and that entirely meets customer requirements.

However, two main things can be noticed:

1. Both people and quality management lack of a strategic vision.
2. People and quality are still treated as not related, despite the intuitive connection.

8.5.2 Lack of a strategic vision in both people and quality management

People management doesn’t involve a plan for the whole organization’s growth. Plans concern only what the employee wants to do and a very few is said about what the company wants its employees to become over time, which competences companies wish to acquire and which level of maturity teams are expected to achieve.

Quality management lacks of both a clear approach to quality (what quality is and how to measure it in a structured, consistent way) and of a long-term strategy (what quality companies achieve now, where they want to go from here to one, two or five years).

8.5.3 People and quality are believed to be related, but the relationship is not directly managed

The relationship between individual competences and team skills and software quality is believed to exist, but the primary focus of managerial efforts is still on processes. It's taken for granted that they way the workforce is handled can lead to better products, but the influence of people over quality is not directly managed.

Beside everyday informal monitoring by the manager, the relationship between people and quality is indirectly assessed and managed through collective meetings scheduled according to the development plan (e.g. Sprint Retrospective) and one-to-one meetings that take place when needed (e.g. when an employee is spotted underperforming).

The value of this indirect management through indirect monitoring is not to be underestimated: the sooner problems, difficulties or "unhappiness" are detected, the higher is the chance that quality is improved, as faults and re-work are reduced, competences are exploited more effectively, and motivation is enhanced.

8.5.4 Establishing a path for a direct management of the people-quality relationship

In order to have this relationship directly and explicitly managed, it's important to establish a path for this management to be done.

Very little has been said in the literature about this, therefore the following directions are prone to further substantial improvements. Still they aim to provide a practical guide for managers to approach in a structured and systematic way a concept (i.e. the relationship between people and quality) that has been taken for granted until now.

Step 1: Purpose definition

The purpose of a direct management can be gathered from quoting once again the interviewed managers:

"The most important thing for good quality software is to have good quality employees" (Magnus Werner, SAAB Training and Simulation).

"*The quality of the product depends on whether the people that develop are feeling good" (Anna Leo, Swedish Board of Agriculture).*

Therefore, managerial actions undertaken should aim to:

1. enhance the quality of the company’s employees, which means the level of their competences and team skills
2. build a productive environment, where people are put in favourable conditions to do their job at the best.

Direct management means this kind of decisions should be made according to an explicitly defined strategy that points out the exact level of quality to be achieved.

Step 2: Organizational audit

The current situation should be depicted:

- Knowledge, competences and team skills. Which approach to knowledge management has been chose? Which strategy has been selected for managing and sharing knowledge? Which kind of IT support has been provided? What kind of competences are available within the company? Where do these competences can be retrieved (i.e. which people embody which competences)?
- Processes and environment. Which methodologies for process management are implemented? Which benefits and which issues do they bring to software people? Which practices do developers refer to? Why have these practices been chosen? What’s the level of maturity of the company’s teams? Are the developers satisfied (“happy”)? Are they motivated?
- Quality. How is quality defined? How is it measured? What’s the current level of quality? Which expectations does the customer have?

Step 3: SWOT analysis

Strengths, weaknesses, opportunities and threats should be drawn from the organizational audit: What does the company do well? What can be done better? What has not been done but should be? What is done and should not be?

In particular: Is there a clear approach to knowledge management? Is the way knowledge is shared appropriate? Is the IT support good enough? Are the competences available effectively exploited? Does the company need other competences? Do the methodologies and practices implemented for software development bring substantial benefits?

Answering to these questions may help managers to find out what can be improved.

Step 4: Objectives and issues

As the company may not have enough resources (or may not want) to improve everything, choices should be made.

Questions to be asked: Which knowledge and competences does the company want to acquire? Which level of maturity does the company want teams to achieve? Which level of workforce “happiness” does the company aim to meet? Which level of product quality does the company strives for providing?

Step 5: Strategy definition and implementation

In order to manage the relationship between people and quality, the whole organizational framework must be clear – that’s why the previous steps should not be skipped.

As pointed out in the beginning of this section, software companies’ people and quality management appear to lack of a strategic vision and the relationship between people and quality is not directly managed. In this step these problems are to be solved.

An organizational-growth plan is to be done: it should say where the company wants to go from now to one / two / five years (in terms of competences owned, maturity of the teams, efficiency of the methodologies applied for software development, quality of the product and customer satisfaction), and – most important – how it wants to do that (which choices are made) and when the improvements are supposed to be carried out.

Planning and carrying out the plan is not enough: every choice made should be carefully assessed in order to find out whether it has brought benefits in terms of quality or not.

Helpful tools for monitoring and evaluating what is done are mentioned within the previous sections (8.2 Knowledge management, 8.3 People and processes management, 8.4 Quality management) and more about these tools can be retrieved within Chapter 2, Chapter 3, and Chapter 4. This follow-up is essential for ensuring that quality is improved: poor managerial choices should be recognized as soon as possible.

9 Conclusions

9.1 Summary of the results

The present work aims to focus the attention of both managers and researchers on the theme of people and quality management. As previously discussed, two main problems have been detected: the lack of a strategic vision in both people and quality management, and the absence of a direct management of the relationship between people and quality. It's believed that software companies can solve these problems through adopting a structured approach to the whole company's management.

In order to support a systematic approach, a simple and practical guide has been provided at the end of Chapter 8. The guide is very concise, but each step relies on a set of concepts that have been elaborated in this thesis. In particular steps 1 – 4 refer to typical managerial activities, aim to provide managers with a well-defined picture of the current situation (focusing on what in the final analysis has an effect over quality) and help to better understand what's the overall goal; the fifth step is concerned with actually solving the two problems pointed out before through using the tools described in this thesis to define a plan for the whole organization's growth, carry it out, and carefully monitor and evaluate it in order to improve quality.

9.2 Limitations

The number of interviewed and surveyed companies was limited: even though statistics may not be that interesting, a wider collection of data and information could be very helpful for a better understanding of the reality of software companies.

The directions provided in section 8.5.4 can be markedly elaborated and deepened. Through a greater number of interviews, more can be suggested about best practices, useful methodologies, and assessment processes.

9.3 Further research

This work has pointed out that there is much more to explore and study. It would be very interesting to do quantitative researches spread over time to investigate on the field:

- How competences and team skills quantitatively influence the outcome of the development process (in order to validate the results of Vinod et al. [63]). This would encourage the evaluation of skills and competences and lead to a better understanding on their effect over quality.

- How choices in people and processes management can lead to concrete changes in terms of quality (e.g. Would the introduction of an index of “who knows what” available to everyone reduce the amount of time required to complete the product? Does the choice of having people working together in an open office improve the level of maturity of the team and does the improved maturity lead to a reduction of the number of errors?).

10 References

[1] Bach J., 1995, *Enough about process: what we need are heroes*, IEEE Software, 12 (2), pg.96-98.

[2] Basili V. R., Weiss D. M., 1984, *A Methodology for Collecting Valid Software Engineering Data*, IEEE Transactions on Software Engineering, SE-10 (6), pg.728-738.

[3] Bass B. M., Dunteman G., 1963, *Behaviour in groups as a function of self interaction and task orientation*, Journal of Abnormal and Social Psychology, 66, pg.419-428.

[4] Bevan N., 1997, *Quality in use: incorporating human factors into the software engineering lifecycle*, 3rd International Software Engineering Standards Symposium.

[5] Boehm B. et al., 1978, *Characteristics of Software Quality*, Elsevier North-Holland Publishing Company.

[6] Borkan J.M., 2004, *Mixed Methods Studies: A Foundation for Primary Care Research,* Annals of Family Medicine, 2, pg.4-6.

[7] Bourque L. B., Fielder E. P., 2003, *How to Conduct Self-Administered and Mail Surveys 2nd edition,* SAGE Publications.

[8] Byrman A., 2006, *Integrating quantitative and qualitative research: how is it done?*, SAGE Publications, 6 (1), pg.97-113.

[9] Canavor N., Meirowitz C., 2010, *How to Interview Effectively*, FT Press Delivery Elements.

[10] Crosby P. B., 1979, *Quality Is Free: The Art of Making Quality Certain*, McGraw-Hill.

[11] Curtis B., Hefley B., Miller S., 2009, *People Capability Maturity Model (P-CMM) Version 2.0, Second Edition,* Carnegie Mellon, pg.1-16.

[12] Ivi, pg.17-28.

[13] Denning P. J., 1992, *What is Software Quality?*, A Commentary from Communications of ACM.

[14] Fink A., 2003, *The Survey Handbook 2nd edition*, SAGE Publications.

[15] Gable, G. G., 1994, *Integrating case study and survey research methods: an example in information systems,* European Journal of Information Systems, 3 (2), pg.112-126.

[16] Garvin D. A., 1984, *What does "product quality" really mean?*, Sloane Management Review.

[17] Ghauri P., Gronhaug K., Kristianslund I., 1995, *Research Methods in Business Studies: A Practical Guide*, Prentice Hall, Hemel Hempstead.

[18] Gillham B., 2008, *Small-Scale Social Survey Methods,* Continuum International Publishing Group.

[19] Greene J.C., Caracelli V.J., Graham, W.F., 1989, *Toward a Conceptual Framework for Mixed-method Evaluation Designs,* Educational Evaluation and Policy Analysis, 11 (3), pg.255-274.

[20] Gubrium J. F., Holstein J.A., 2002, *Handbook of Interview Research. Context and Method,* SAGE Publications.

[21] Hahn J., Subramani M.R., 2000, *A framework of knowledge management systems: issues and challenges for theory and practice,* ICIS '00 Proceedings of the twenty first international conference on Information systems.

[22] Hansen M., Nohria N., Tierney T., 1999, *What's your strategy for managing knowledge?*, Harvard Business Review.

[23] Hedlund G., 1994, *A model of knowledge management and the N-form corporation*, Strategic Management Journal, 15, pg.73-90.

[24] Hellström T., Malmquist U., Mikaelsson J., 2001, *Decentralizing knowledge. Managing knowledge work in a software engineering firm*, Journal of High Technology Management Research, 12, pg.25-38.

[25] Herbsleb J., Zubrow D., Goldenson D., Hayes W., Paulk M., 1997, *Software Quality and the Capability Maturity Model*, Communications of the ACM, 40 (6).

[26] Hines A. M., 1993, *Linking Qualitative and Quantitative Methods in Cross-Cultural Survey Research: Techniques from Cognitive Science*, American Journal of Community Psychology, 2L, 6.

[27] Humphrey W. S., 1989, *Managing the Software Process,* Addison-Wesley Professional.

[28] Humphrey W. S., 1995, *Introducing the personal software process*, Annals of Software Engineering 1, pg.311-325.

[29] Jick T. D., 1979, *Mixing Qualitative and Quantitative Methods: Triangulation in Action,* Administrative Science Quarterly, 24 (4), pg.602-611.

[30] Junnarkar B., 1997, *Leveraging Collective Intellect by Building Organizational Capabilities*, Expert Systems with Applications, 13, pg.29-40.

[31] Kaplan B., Duchon D., 1988, *Combining Qualitative and Quantitative Methods in Information Systems Research: A Case Study*, MIS Quarterly, 12 (4), pg.571-586.

[32] Maslow A. A., 1954, *Motivation and Personality,* Harper and Row.

[33] Miles G., Miles R. E., Perrone V., Edvinsson L., 1998, *Some conceptual and research barriers to the utilization of knowledge*, California Management Review, 40 (3), pg.281-288.

[34] Musa J.D., Iannino A., Okumoto K., 1987, *Engineering and Managing Software with Reliability Measures*, McGraw-Hill.

[35] Nowak M. J., Grantham C. E., 2000, *The virtual incubator: managing human capital in the software industry,* Research Policy, 29, pg.125-134.

[36] O'Dell C., Grayson C.J., 1998, *If Only We Knew What We Know: identification and transfer of internal best practices*, California Management Review.

[37] Pfeffer J., 1994, *Competitive Advantage through People*, Harvard Business School Press.

[38] Plano Clark V. L., 2010, *The Adoption and Practice of Mixed Methods: U.S. Trends in Federally Funded Health-Related Research*, Qualitative Inquiry, 16, pg.428-440.

[39] Prahalad C. K., Hamel G., 1990, *The Core Competence of the Corporation*, Harvard Business Review 68, 3, pg.79-91.

[40] Prusak L., Cohen D., 1998, *Knowledge buyers, sellers and brokers: the political economy of knowledge* in Neef D., Siesfeld A. G, Cefola J., 1998, *The Economic Impact of Knowledge,* Butterworth Heinemann, pg.137-159.

[41] Sarvary M., 1999, *Knowledge management and competition in the consulting industry*, California Management Review, 41 (2), pg.95-107.

[42] Schreiber G., Akkermans H., Anjewierden A., De Hoog R., Shadbolt N., Van de Velde W., Wielinga B., 2000, *Knowledge Engineering and Management: The CommonKADS Methodology*, MIT Press, pg.2.

[43] Ivi, pg.4.

[44] Ivi, pg.6.

[45] Ivi, pg.13-22.

[46] Ivi, pg.70-71.

[47] Ivi, pg.75-79.

[48] Ivi, pg.79-82.

[49] Sommerville I., 2007, *Software Engineering 8th Edition*, Pearson Education, pg.74-81.

[50] Ivi, pg.591-592.

[51] Ivi, pg.593-595.

[52] Ivi, pg.599-606.

[53] Ivi, pg.607-609.

[54] Sommerville I., 2010, *Software Engineering 9th Edition*, Pearson Education, pg.652-654.

[55] Ivi, pg.655-657.

[56] Ivi, pg.657-662.

[57] Ivi, pg.663-667.

[58] Ivi, pg.668-677.

[59] Stone D. H., 1993, *How to do it. Design a questionnaire,* British Medical Journal, 307, pg.1264-1265.

[60] Thomas S. J., 1999, *Designing Surveys That Work! A Step-by-Step Guide*, Corwin Press, Inc.

[61] Tsoukas H., 1996, *The firm as a distributed knowledge system: a constructionist approach*, Strategic Management Journal, 17, pg.11-25.

[62] Vidich A. J., Shapiro G., 1955, *A Comparison of Participant-Observation and Survey Data*, American Sociological Review, 20, pg.28-33.

[63] Vinod V., Dhanalakshmi J., Sahadev S., 2009, *Software Team Skills on Software Product Quality,* Asian Journal of Information Technology, 8 (1), pg.8-13.

[64] Webb J., 1992, *Understanding and Designing Marketing Research*, The Dryden Press.

[65] Wielinga B., Sandberg J., Schreiber G., 1997, *Methods and Techniques for Knowledge Management: What Has Knowledge Engineering to Offer?*, Expert Systems with Applications, 13, pg.73-84.

[66] Wiig K.M., De Hoog R., Van der Spek R., 1997, *Supporting Knowledge Management: A Selection of Methods and Techniques*, Expert Systems with Applications, 13, pg.15-27.

[67] Yin R. K., 1984, *Case Study Research: Design and Methods*, SAGE Publications.

[68] Agile Manifesto, http://agilemanifesto.org/ (2011-05-26)

[69] Agile Modeling, http://www.agilemodeling.com/ (2011-05-26)

[70] Engineering and Managing Knowledge, CommonKADS http://www.commonkads.uva.nl/ (2011-04-08)

[71] Hoffman D., The Darker Side of Metrics, http://www.softwarequalitymethods.com/Papers/DarkMets%20Paper.pdf (2011-04-10)

[72] Kaner C., Software Engineering Metrics: What Do They Measure and How Do We Know?, http://www.kaner.com/pdfs/metrics2004.pdf (2011-04-10)

[73] Scrum, http://www.scrum.org/ (2011-05-26)

[74] Software Engineering Institute, People CMM, http://www.sei.cmu.edu/cmmi/tools/peoplecmm/ (2011-04-10)

[75] Value Based Management, People CMM, http://www.valuebasedmanagement.net/methods_people_capability_maturity_model.html (2011-04-10)

[76] Wikipedia, Agile software development, http://en.wikipedia.org/wiki/Agile_software_development (2011-05-26)

[77] Wikipedia, Extreme Programming, http://en.wikipedia.org/wiki/Extreme_Programming (2011-05-26)

[78] Wikipedia, ISO/IEC 15504, http://en.wikipedia.org/wiki/ISO/IEC_15504 (2011-04-10)

[79] Wikipedia, ISO/IEC 9126, http://en.wikipedia.org/wiki/ISO/IEC_9126 (2011-04-10)

[80] Wikipedia, Knowledge management, http://en.wikipedia.org/wiki/Knowledge_ management (2011-04-08)

[81] Wikipedia, RUP, http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process (2011-05-26)

[82] Wikipedia, Scrum, http://en.wikipedia.org/wiki/Scrum_(development) (2011-05-26)

[83] Wikipedia, Software quality, http://en.wikipedia.org/wiki/Software_quality (2011-04-10)

[84] Shaukat A., Research Methodology, http://www.journal.au.edu/abac_journal/jan98/article5.html (2011-04-10)

[85] SQA, ISO 9126 Software Quality Characteristics, http://www.sqa.net/iso9126.html (2011-04-10)

[86] Zrymiak D., People – Capability Maturity Model, http://www.msi.ms/MSJ/People-Capability_Maturity_Model.htm (2011-04-10)

11 Appendix

Appendix 11.1 presents the cover letter attached to the questionnaire.

Appendix 11.2 presents the questions asked in the survey.

Appendix 11.3 collects screenshots of how the questionnaire looks like to the respondent.

Appendix 11.4 presents an overview on Agile software development and on some of the practices the interviewed managers have referred to.

11.1 Cover letter

Dear Sir / Madam,

My name is Elena and I’m an Italian student. I have a bachelor degree in Engineering and Management from the University of Padua and by the end of 2011 I’ll take my master degree.

I’m currently an exchange student at Jönköping University (Sweden) and I’m working on my master project, which investigates how the management of individuals in a software company can be addressed to positively affect the quality of the software product. The basic idea is that the better people are managed, the better is the quality of the product and, consequently, the more the customer is satisfied.

The purpose of the attached questionnaire, which is part of my project, is to collect information on the management of software people and of software quality in your company.

Your participation in this study is expected to lead to a better understanding of how individual competencies and software quality are managed in some real cases and how much companies reckon that effectively managing people affects the quality of their software products.

Compiling the questionnaire will take approximately 20 minutes. It would be greatly appreciated if you could complete the questionnaire by May 7th. All data collected in this questionnaire will be treated confidentially. The results of this study will be presented in my graduation thesis, which will be available online (I’ll provide you a link as soon as the thesis is published).

I am very thankful for your willingness to fill in the form. If you have any question, please contact soel10cd@student.hj.se.

Yours faithfully,

Elena Sortino

Elena Sortino
School of Engineering
Jönköping University
Jönköping, Sweden
Tel: +460765941593
E-mail: soel10cd@student.hj.se

11.2 Questions

11.2.1 PART 1: How do you manage knowledge?

1. In which percentage would you estimate the intellectual capital of your company to be responsible for the total added value of your products?

< 50%

50% – 65%

65% – 80%

> 80%

2. Is there an explicit predominant strategy for knowledge management in your company?

No, there isn't any strategy. People are free to share knowledge as they prefer.

Yes, a people-to-document approach is used. Knowledge is extracted and codified from the person who developed it, so that any other person can search for it without having to contact the original owner.

Yes, a network of people is defined. Knowledge is shared through interpersonal communications and is transferred through a connection between individuals (brainstorming sessions, one-on-one conversation, telephone, e-mail, job rotation, etc.).

3. Is there any kind of dedicated IT support for knowledge sharing in your company?

No, there is no particular tool. People use common tools (telephone, e-mail, meetings, etc.) as the need of knowledge sharing arises.

No, there is no particular tool, but people use common tools (telephone, e-mail, meetings, etc.) in a standardized way, depending on the particular need.

Yes, there is a dedicated IT system for knowledge sharing (online library / electronic repository / database / forum / …). [go to 3.1]

3.1. Whether in your company there is a dedicated IT system for knowledge sharing, please describe it with few words.

4. Is knowledge sharing explicitly encouraged and rewarded in your company?

No, knowledge sharing is not a managerial issue.

Yes, knowledge sharing is explicitly encouraged and rewarded. [go to 4.1]

Knowledge sharing is explicitly encouraged, but not rewarded. [go to 4.1]

4.1. Whether in your company knowledge sharing is explicitly encouraged and – eventually – rewarded, please give at least an example of how this happens.

5. Is there any explicit methodology you apply for managing knowledge?

No, we don’t apply any methodology.

Yes, we refer to the knowledge-management cycle (Review-Conceptualize-Reflect-Act).

Yes, we apply the CommonKADS methodology (defining a knowledge analysis framework based on organization, task, agent, knowledge, communication and design models).

Yes, we apply … [see 5.1]

5.1. Whether the methodology in use for managing knowledge is not present in the suggested answers, please describe it here with few words.

11.2.2 PART 2: How do you manage software development and software people?

6. Is there any explicit methodology you apply for managing, assessing and improving the software development process?

No, we don't apply any methodology.

Yes, we refer to the SPICE (Software Process Improvement and Capability Determination) framework.

Yes, we apply the CMMI (Capability Maturity Model Integration) methodology.

Yes, we apply … [see 6.1]

6.1. Whether the methodology in use for managing the software development process is not present in the suggested answers, please describe it here with few words.

7. Is there any explicit methodology you apply for managing, assessing and improving the software workforce?

No, we don't apply any methodology.

Yes, workforce takes the PSP (Personal Software Process) course.

Yes, we apply the P-CMM (People - Capability Maturity Model) methodology.

Yes, we apply … [see 7.1]

7.1. Whether the methodology in use for managing the software workforce is not present in the suggested answers, please describe it here with few words.

8. Whether a methodology for improving the workforce capabilities has been applied, which results has it brought? Please, describe them here.

11.2.3 PART 3: How do you manage software quality?

9. Is there any explicit approach to the management, assessment and improvement of the software quality?

No, we don’t have a systematic approach to software quality.

Yes, we approach quality according to standard definitions. [go to 9.1]

Yes, we approach quality according to the user’s perception. [go to 9.2]

Yes, we approach quality according to customer satisfaction. [go to 9.3]

Yes, we approach quality according to the software lifecycle. [go to 9.4]

Yes, we approach quality according to … [see 9.5]

9.1. Please describe with few words the standard you refer to.

9.2. Please, describe your approach to quality from the perspective of the user’s perception.

9.3. Please, describe your approach to quality from the perspective of customer satisfaction.

9.4. Please, describe your approach to quality from the perspective of the software lifecycle.

9.5. Whether the approach in use is not present in the suggested answers, please describe it here.

10. Do you believe that the quality of your software depends on how you manage the people who develop it?

No, I don’t. [skip to PART 4]

Yes, I believe that the way we manage our people affects the quality of the product.

11.2.4 People management and software quality

11. Please, describe the tools (surveys, interviews, observation, statistics, …) you use – or you're planning to use – to evaluate the relationship between the way software people are managed and product quality.

11.1. Which results have emerged from your evaluation?

12. Please, give at least an example of a strategic decision over people management that has affected the quality of your product. [More than one example will be appreciated].

13. Please, describe what have you planned for the future in order to improve the level of your software people' skills and competencies (seminars, courses, training, coaching, hire more qualified people…).

13.1 How do you think these improvements will enhance the quality of your product (software quality attributes, standard conformity, customer satisfaction rate, sales, lead times, …)?

11.2.5 PART 4: General information

The respondent works in…

Software house

IT department of a service company

IT department of a manufacturing company

IT department of a no-profit organization

IT department of a public agency

HR department of a service company

HR department of a manufacturing company

HR department of a no-profit organization

HR department of a public agency

The size of the company / department is…

micro (< 10 people)

small (< 50 people)

medium sized (< 250 people)

large (> 250 people)

The company is located in…

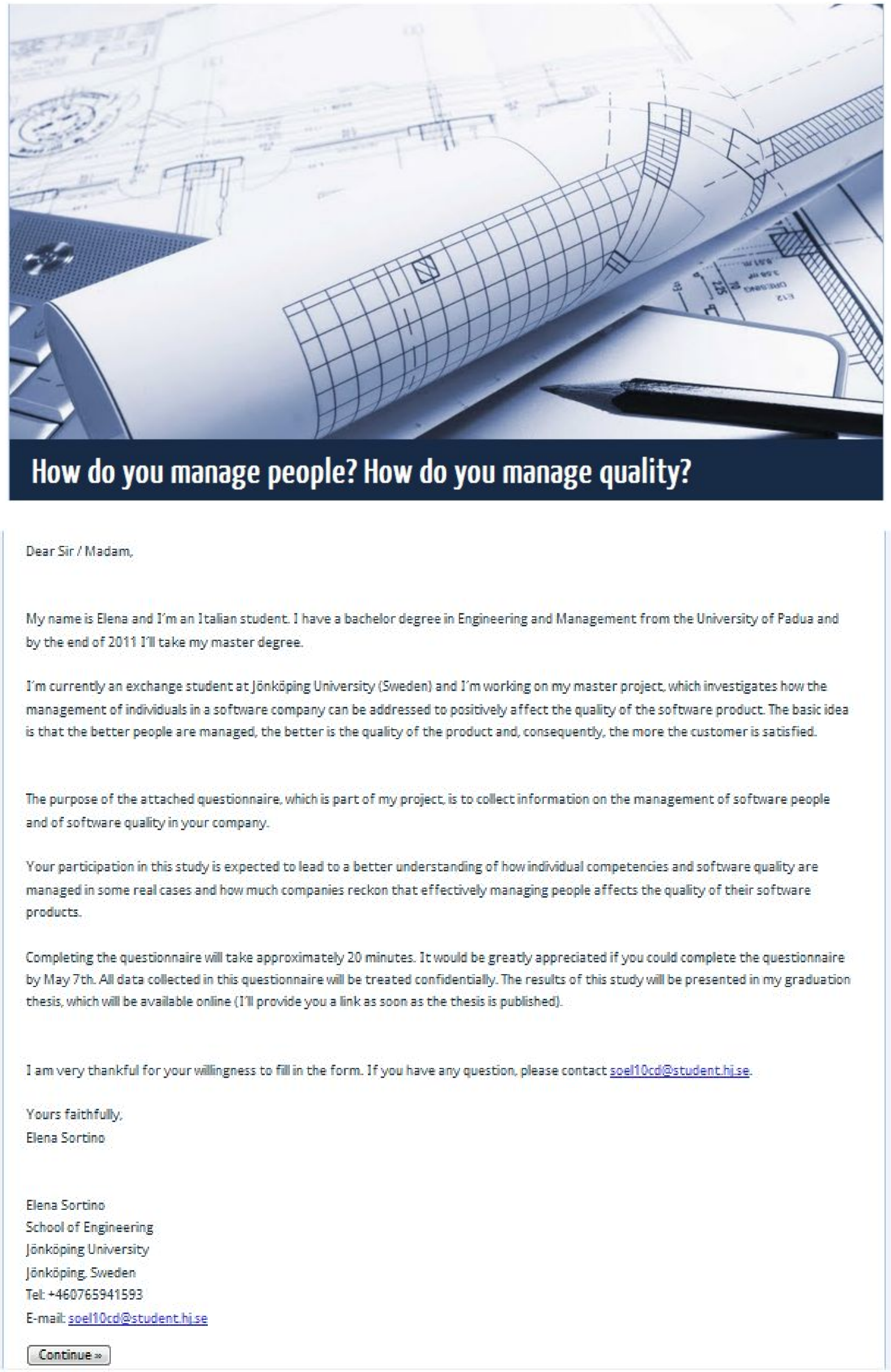
Sweden

Italy

Other country

OPTIONAL: Name of the company, Contacts for further information (name of the respondent, e-mail, telephone…).

11.3 Screenshots



How do you manage people? How do you manage quality?

Dear Sir / Madam,

My name is Elena and I'm an Italian student. I have a bachelor degree in Engineering and Management from the University of Padua and by the end of 2011 I'll take my master degree.

I'm currently an exchange student at Jönköping University (Sweden) and I´m working on my master project, which investigates how the management of individuals in a software company can be addressed to positively affect the quality of the software product. The basic idea is that the better people are managed, the better is the quality of the product and, consequently, the more the customer is satisfied.

The purpose of the attached questionnaire, which is part of my project, is to collect information on the management of software people and of software quality in your company.

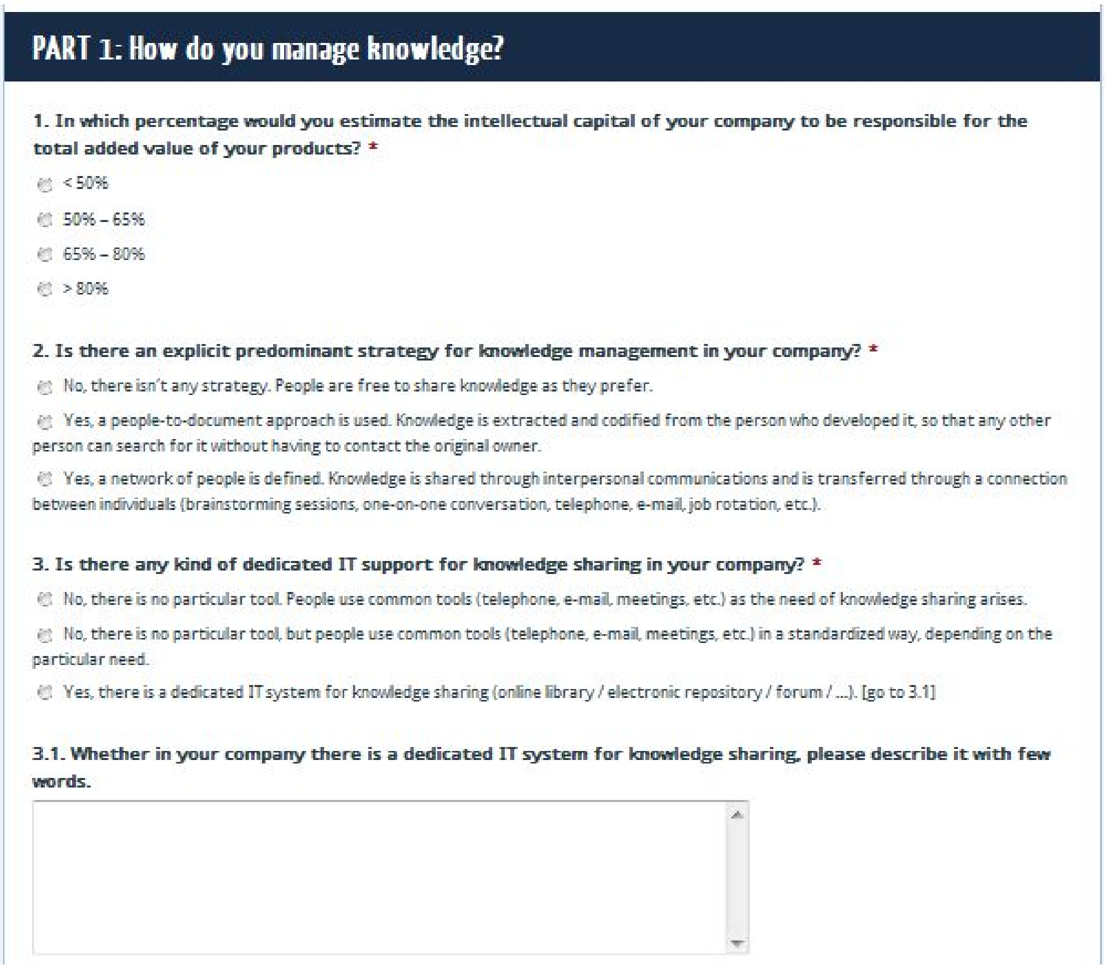
Your participation in this study is expected to lead to a better understanding of how individual competencies and software quality are managed in some real cases and how much companies reckon that effectively managing people affects the quality of their software products.

Completing the questionnaire will take approximately 20 minutes. It would be greatly appreciated if you could complete the questionnaire by May 7th. All data collected in this questionnaire will be treated confidentially. The results of this study will be presented in my graduation thesis, which will be available online (I'll provide you a link as soon as the thesis is published).

I am very thankful for your willingness to fill in the form. If you have any question, please contact soel10cd@student.hj.se.

Yours faithfully,
Elena Sortino

Elena Sortino
School of Engineering
Jönköping University
Jönköping, Sweden
Tel: +460765941593
E-mail: soel10cd@student.hj.se



PART 1: How do you manage knowledge?

1. In which percentage would you estimate the intellectual capital of your company to be responsible for the total added value of your products? *

○ < 50%

○ 50% – 65%

○ 65% – 80%

○ > 80%

2. Is there an explicit predominant strategy for knowledge management in your company? *

○ No, there isn't any strategy. People are free to share knowledge as they prefer.

○ Yes, a people-to-document approach is used. Knowledge is extracted and codified from the person who developed it, so that any other person can search for it without having to contact the original owner.

○ Yes, a network of people is defined. Knowledge is shared through interpersonal communications and is transferred through a connection between individuals (brainstorming sessions, one-on-one conversation, telephone, e-mail, job rotation, etc.).

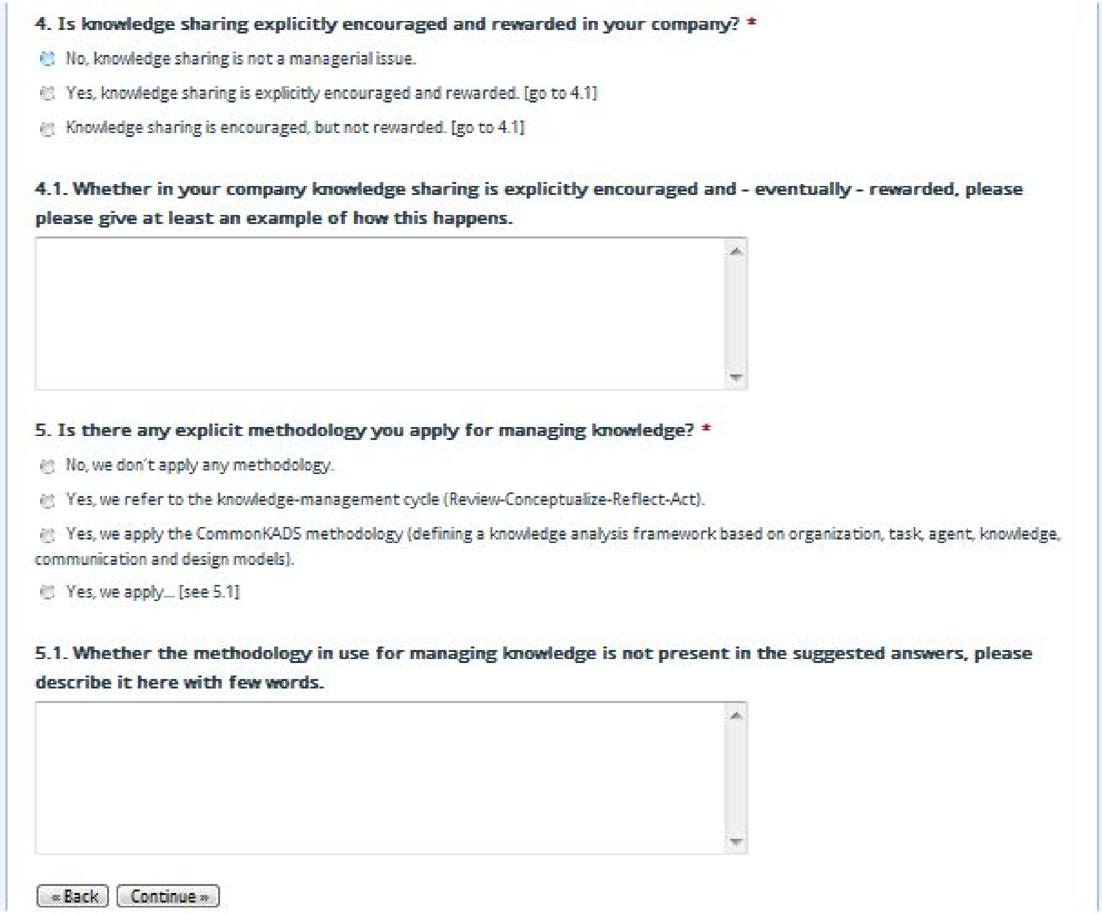
3. Is there any kind of dedicated IT support for knowledge sharing in your company? *

○ No, there is no particular tool. People use common tools (telephone, e-mail, meetings, etc.) as the need of knowledge sharing arises.

○ No, there is no particular tool, but people use common tools (telephone, e-mail, meetings, etc.) in a standardized way, depending on the particular need.

○ Yes, there is a dedicated IT system for knowledge sharing (online library / electronic repository / forum / ...). [go to 3.1]

3.1. Whether in your company there is a dedicated IT system for knowledge sharing, please describe it with few words.



4. Is knowledge sharing explicitly encouraged and rewarded in your company? *

○ No, knowledge sharing is not a managerial issue.

○ Yes, knowledge sharing is explicitly encouraged and rewarded. [go to 4.1]

○ Knowledge sharing is encouraged, but not rewarded. [go to 4.1]

4.1. Whether in your company knowledge sharing is explicitly encouraged and - eventually - rewarded, please please give at least an example of how this happens.

5. Is there any explicit methodology you apply for managing knowledge? *

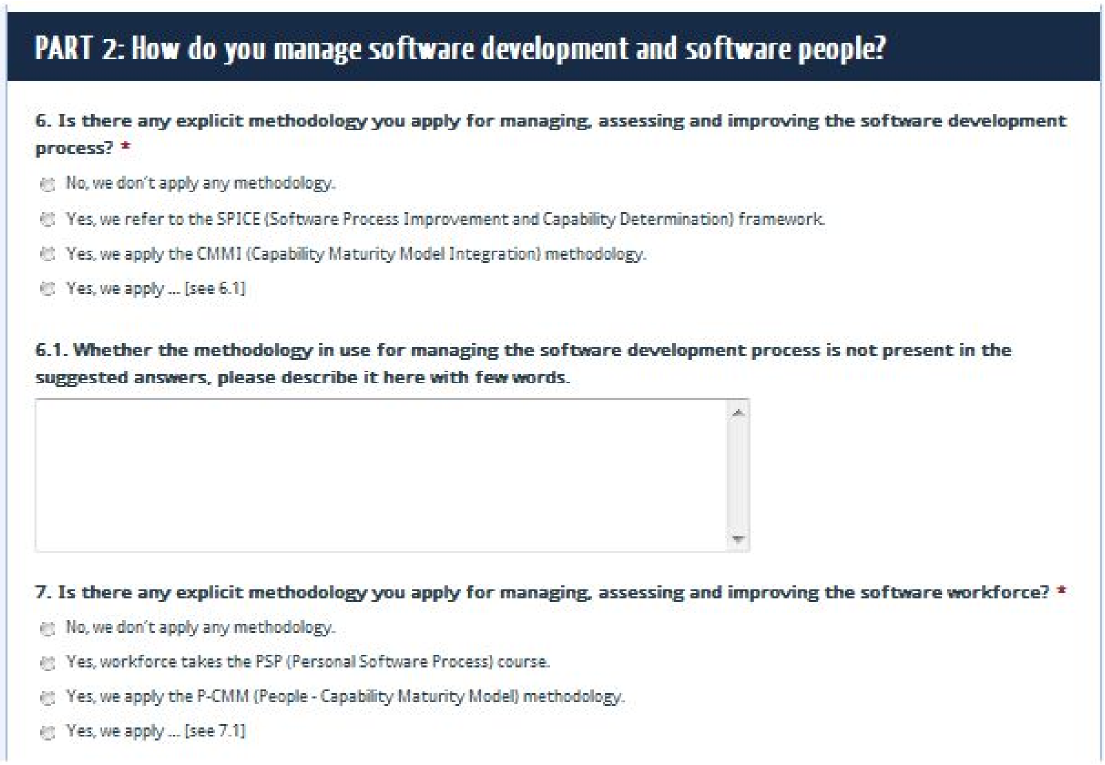
○ No, we don't apply any methodology.

○ Yes, we refer to the knowledge-management cycle (Review-Conceptualize-Reflect-Act).

○ Yes, we apply the CommonKADS methodology (defining a knowledge analysis framework based on organization, task, agent, knowledge, communication and design models).

○ Yes, we apply... [see 5.1]

5.1. Whether the methodology in use for managing knowledge is not present in the suggested answers, please describe it here with few words.



PART 2: How do you manage software development and software people?

6. Is there any explicit methodology you apply for managing, assessing and improving the software development process? *

○ No, we don't apply any methodology.

○ Yes, we refer to the SPICE (Software Process Improvement and Capability Determination) framework.

○ Yes, we apply the CMMI (Capability Maturity Model Integration) methodology.

○ Yes, we apply ... [see 6.1]

6.1. Whether the methodology in use for managing the software development process is not present in the suggested answers, please describe it here with few words.

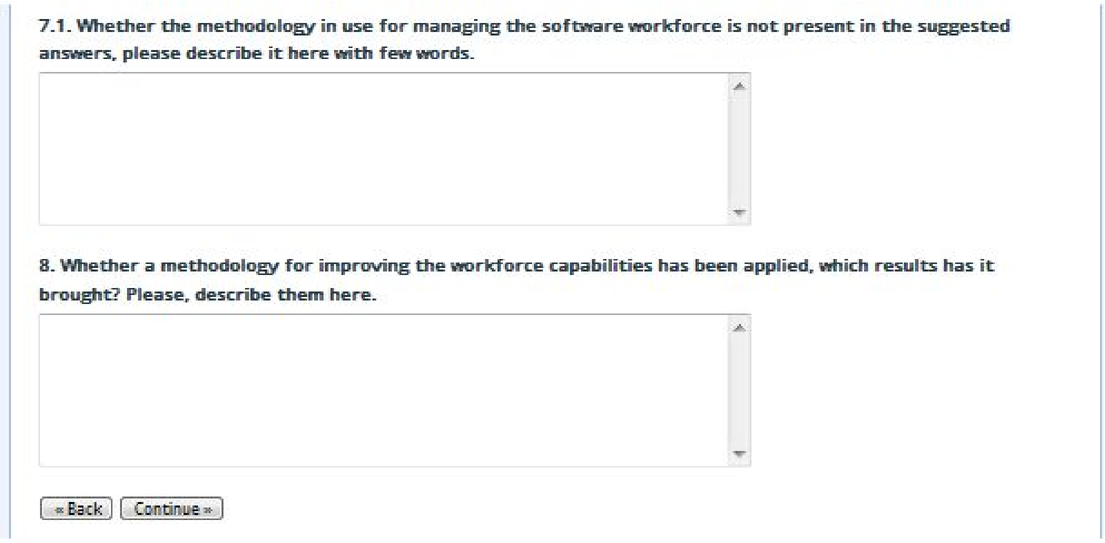
7. Is there any explicit methodology you apply for managing, assessing and improving the software workforce? *

○ No, we don't apply any methodology.

○ Yes, workforce takes the PSP (Personal Software Process) course.

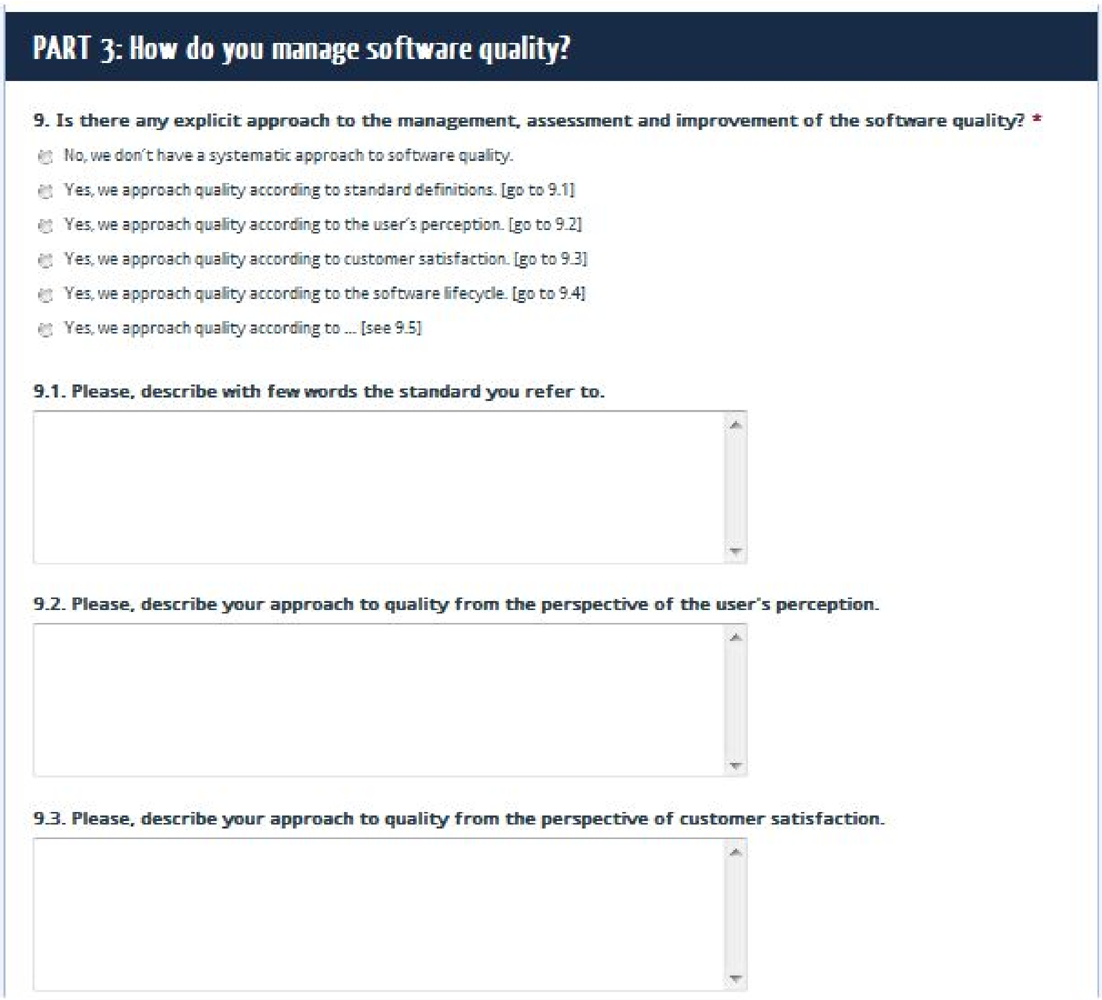
○ Yes, we apply the P-CMM (People - Capability Maturity Model) methodology.

○ Yes, we apply ... [see 7.1]



7.1. Whether the methodology in use for managing the software workforce is not present in the suggested answers, please describe it here with few words.

8. Whether a methodology for improving the workforce capabilities has been applied, which results has it brought? Please, describe them here.



PART 3: How do you manage software quality?

9. Is there any explicit approach to the management, assessment and improvement of the software quality? *

○ No, we don't have a systematic approach to software quality.

○ Yes, we approach quality according to standard definitions. [go to 9.1]

○ Yes, we approach quality according to the user's perception. [go to 9.2]

○ Yes, we approach quality according to customer satisfaction. [go to 9.3]

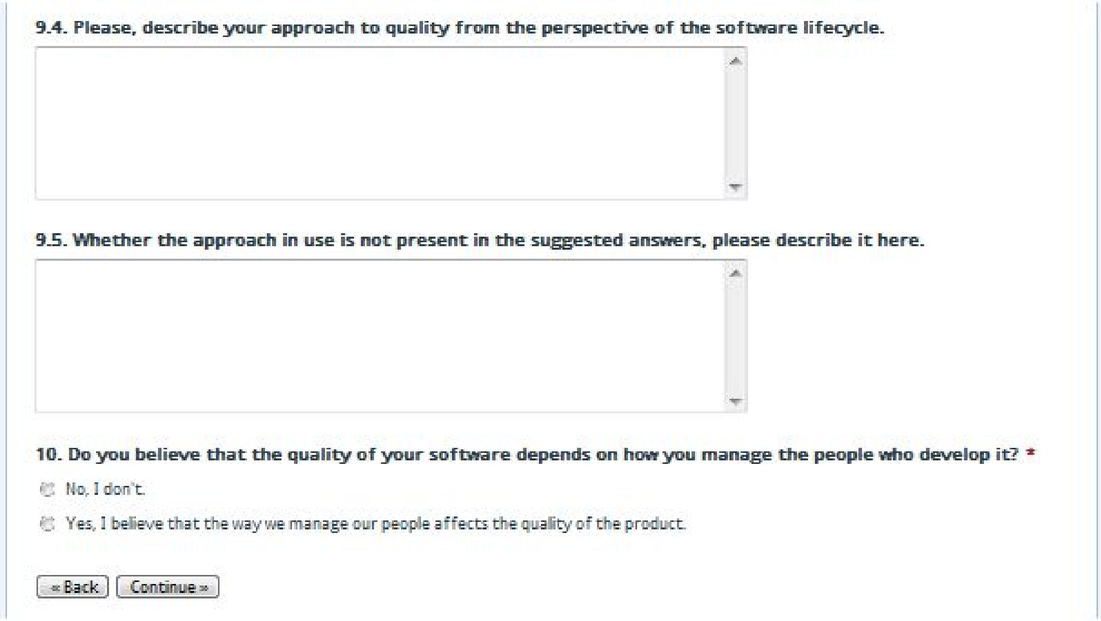
○ Yes, we approach quality according to the software lifecycle. [go to 9.4]

○ Yes, we approach quality according to ... [see 9.5]

9.1. Please, describe with few words the standard you refer to.

9.2. Please, describe your approach to quality from the perspective of the user's perception.

9.3. Please, describe your approach to quality from the perspective of customer satisfaction.



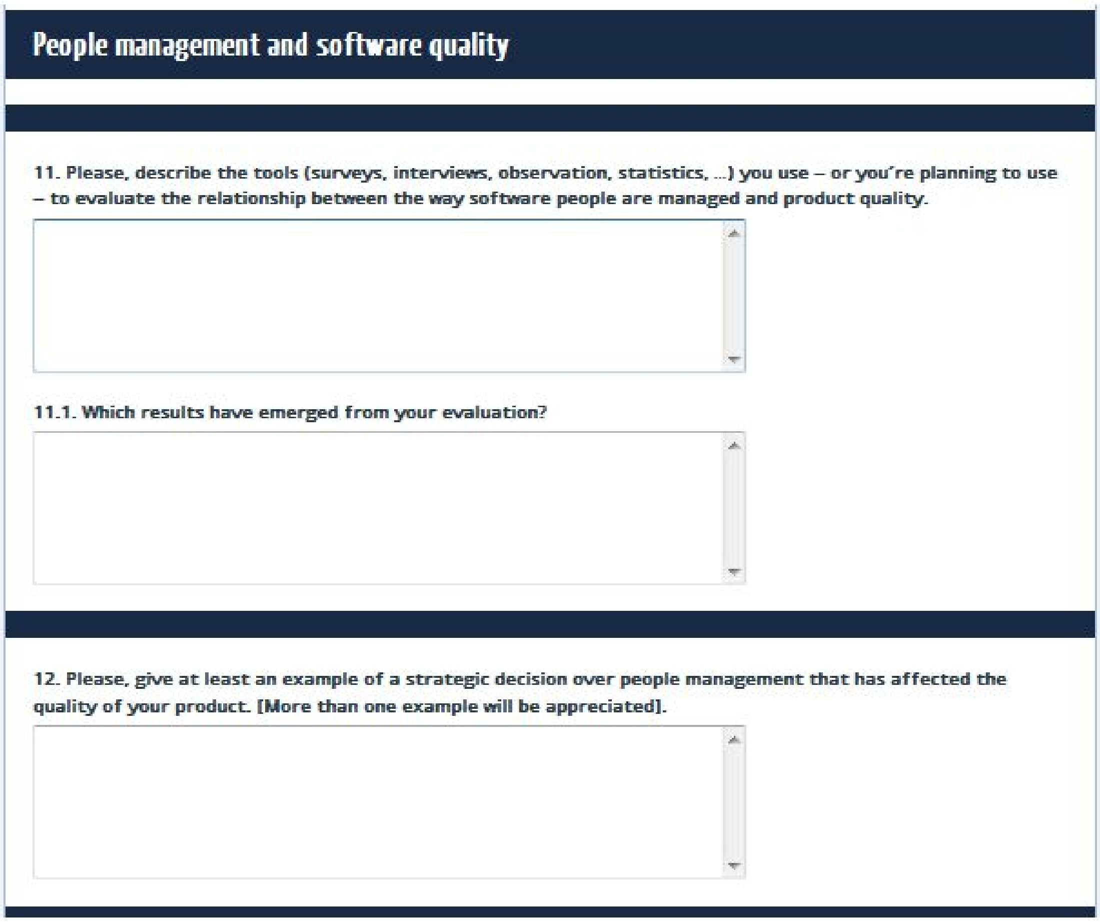
9.4. Please, describe your approach to quality from the perspective of the software lifecycle.

9.5. Whether the approach in use is not present in the suggested answers, please describe it here.

10. Do you believe that the quality of your software depends on how you manage the people who develop it? *

○ No, I don't.

○ Yes, I believe that the way we manage our people affects the quality of the product.

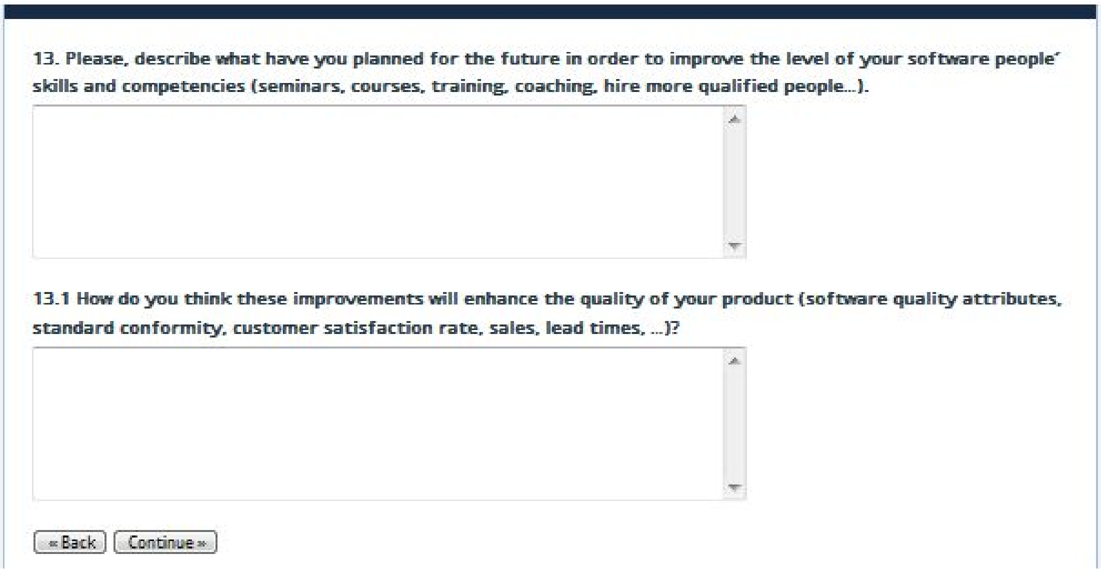


People management and software quality

11. Please, describe the tools (surveys, interviews, observation, statistics, …) you use – or you're planning to use – to evaluate the relationship between the way software people are managed and product quality.

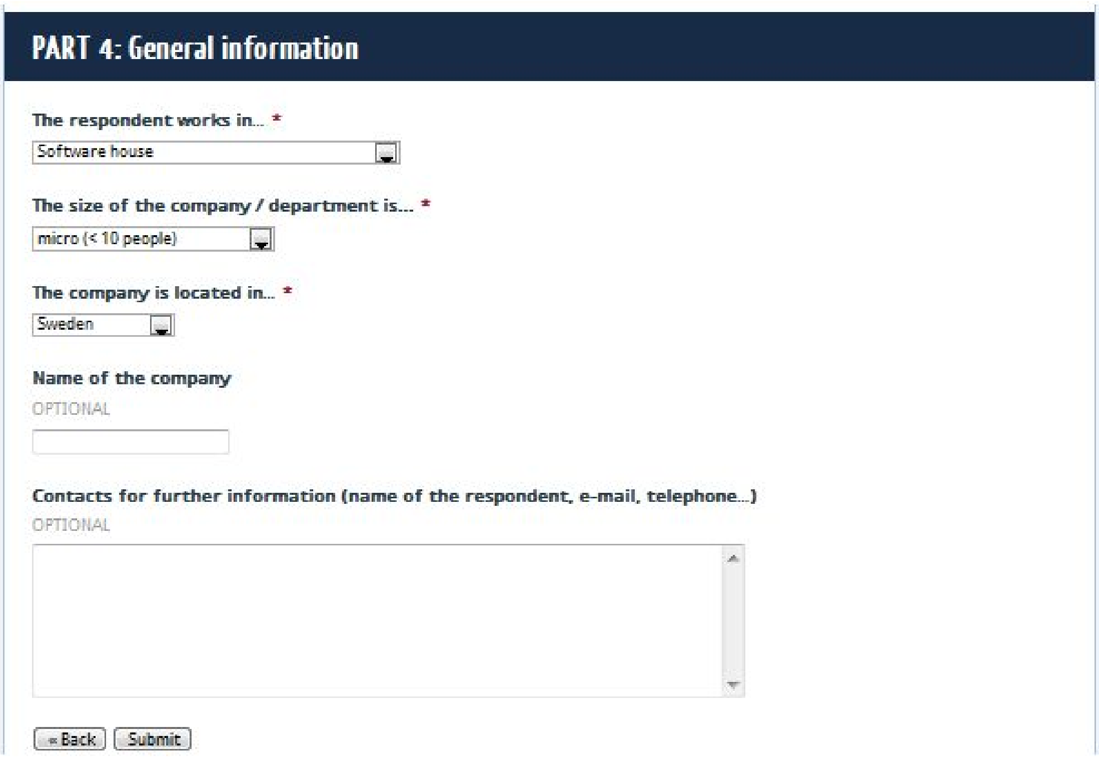
11.1. Which results have emerged from your evaluation?

12. Please, give at least an example of a strategic decision over people management that has affected the quality of your product. [More than one example will be appreciated].



13. Please, describe what have you planned for the future in order to improve the level of your software people´ skills and competencies (seminars, courses, training, coaching, hire more qualified people…).

13.1 How do you think these improvements will enhance the quality of your product (software quality attributes, standard conformity, customer satisfaction rate, sales, lead times, …)?



PART 4: General information

The respondent works in... *

Software house

The size of the company / department is... *

micro (< 10 people)

The company is located in... *

Sweden

Name of the company

OPTIONAL

Contacts for further information (name of the respondent, e-mail, telephone...)

OPTIONAL

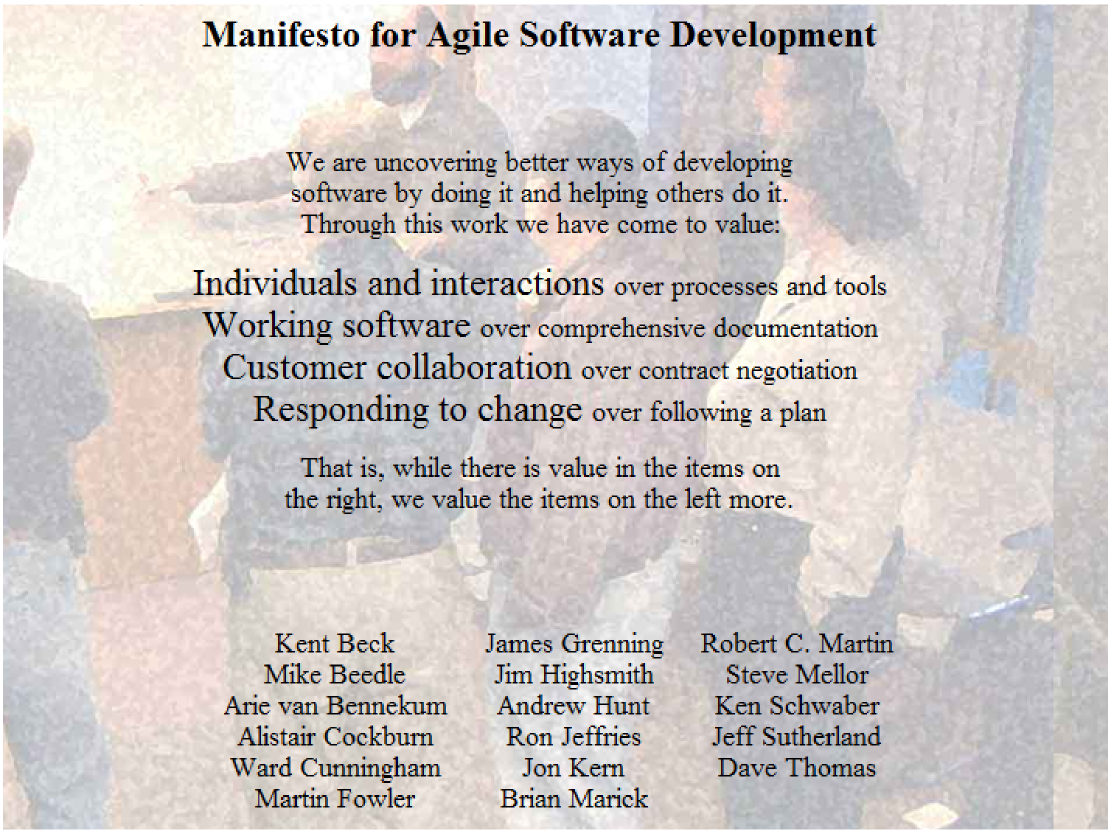
11.4 Agile software development

11.4.1 What does “Agile” mean?

Manifesto for Agile Software Development

Agile software development is a collection of methodologies for an iterative and incremental software development carried out by self-organizing, cross-functional teams [76].

The Agile approach has been defined through the “Manifesto for Agile Software Development” [68] in February 2001. It begins as follows:



Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

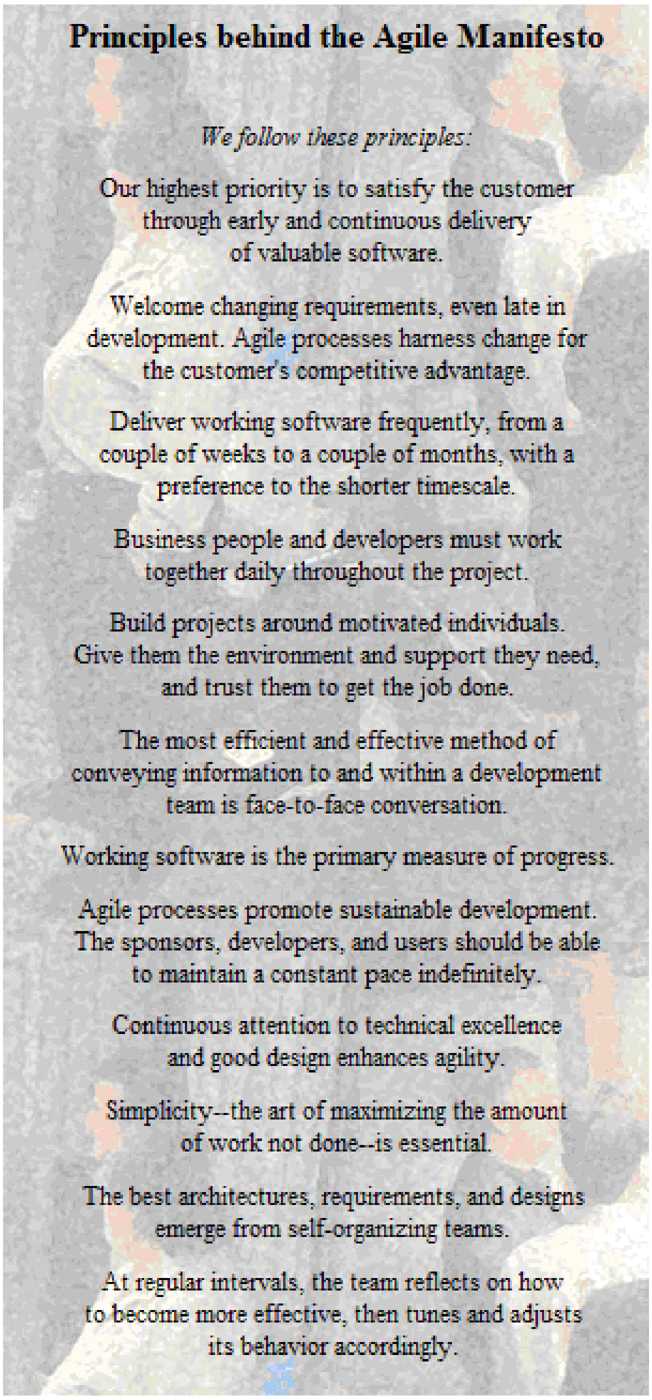
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Fig. 11.1 Manifesto for Agile Software Development [68]

The meaning is that self-organization, motivation and cooperation are important (i.e. “Individual and interactions”), working software is more useful and welcome than presenting papers (i.e. “Working software”), requirements cannot be stated entirely at the beginning of the software development and therefore continuous involvement of customer and stakeholders is required (i.e. “Customer collaboration”), and the focus is on quick responses to change and continuous development (i.e. “Responding to change”).

This Manifesto is based on twelve principles:



Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Fig. 11.2 Principles behind the Agile Manifesto [68]

Agile methods

Agile development methods have in common three main characteristics:

– timeboxed iterations from one to four weeks of full software development cycle (planning, requirement analysis, design, coding, testing and demonstration) to deliver working software

- teamwork, where the team composition is usually cross-functional and self-organizing, and face-to-face communication (often in a single open office)
- customer involvement, as stakeholders and customer representatives work closely with the development team.

Well-known agile methods are: Agile Modeling, RUP (Rational Unified Process) and its simpler version AUP (Agile Unified Process), Scrum, XP (Extreme Programming). They differ in their focus: some focus on the practices (XP, Agile Modeling), others focus on managing the software projects (Scrum), others focus on the development life cycle (RUP) [76].

11.4.2 Agile Modeling

A practice-based methodology

Agile Modeling is a supplement to other agile methodologies such as XP, AUP and Scrum. It’s a practice-based methodology, a collection of values, principles and practices for modeling software development processes and documentation of software-based systems [69].

Best practices

Agile Modeling’s best practices, summarized in **Errore. L'origine riferimento non è stata trovata.**, are [69]:

- Active Stakeholder Participation means that the stakeholders should provide information and make decisions in a timely manner and be actively involved in the development process.
- Architecture Envisioning means that at the beginning of an agile project some kind of high-level architectural modeling should be done to identify a viable technical strategy.
- Document Continuously means writing deliverable documentation throughout the lifecycle.
- Document Late means writing the documentation as late as possible to avoiding speculative ideas.
- Executable Specifications means that requirements should be written in the form of executable "customer tests".

- Iteration Modeling means that at the beginning of each iteration a bit of modeling should be done.
- Just Barely Good Enough (JBGE) means that a model or document should be sufficient for the situation at hand and no more.
- Look Ahead Modeling means that time should be spent on exploring complex requirements.
- Model Storming means that details should be explored on a just-in-time basis during an iteration.
- Multiple Models means that each type of model has it's strengths and weaknesses, so the right model needs to be applied to each situation.
- Prioritized Requirements means that requirements are implemented by the agile team in priority order to provide the greatest return on investment (ROI) possible.
- Requirements Envisioning means that the scope of the project and the initial prioritized stack of requirements need to be identified at the beginning of the project.
- Single Source Information means that information should be captured only in one place.
- Test-Driven Design (TDD) is a just-in-time approach that at the requirements or design level a test should be written and then code should be provided to fulfil that test.

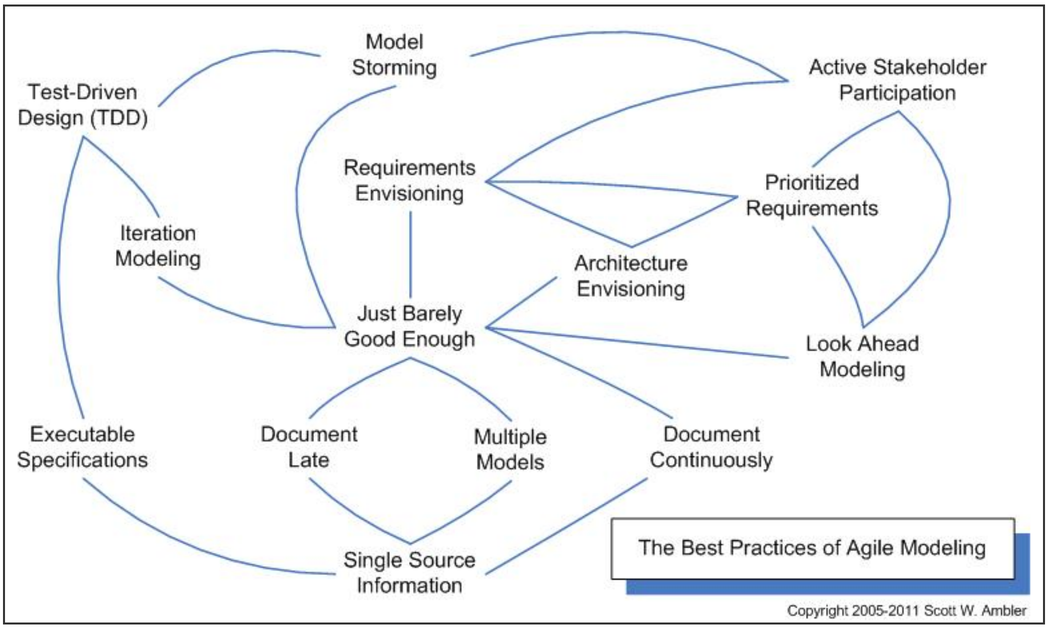


Fig. 11.3 The Best Practices of Agile Modeling [69]

11.4.3 RUP

An iterative software development process framework

The Rational Unified Process is an iterative software development process framework created in 2003 by a division of IBM; it’s not a single process, but a framework that can be tailored by organizations and project teams to their own needs [81].

Even though according to the most common classifications RUP is not an Agile methodology, it has been chosen to report it here as one of them because the interviewed managers appeared to consider it as, in some way, agile (i.e. it’s an iterative framework that can be tailored and implies high customer involvement during the whole development process, unlike, for example, the traditional waterfall model).

Best practices and the RUP concept

Six best practices have been derived from several software companies’ experience with RUP [81]:

- Develop iteratively, with risk as the primary iteration driver (no need to know all requirements in advance)
- Manage requirements (keep in mind what the user wants)
- Employ a component-based architecture (break down complex projects into smaller parts to favour code reuse - for example through the use of OOP, Object-Oriented programming)
- Model software visually, use diagrams to represent components, users and interactions (a useful tool is UML, Unified Modeling Language)
- Continuously verify quality through tests
- Control changes (of team, location, platform, …).

RUP is based on three building blocks: Roles (that define “who”, a set of skills, competencies and responsibilities), Work Products (“what”, a result from a task, including documents and models) and Tasks (“how”, a unit of work assigned to a Role that provides a meaningful result). Tasks may belong to several disciplines: Business Modeling, Requirements, Analysis and Design, Implementation, Test, Deployment, Configuration and Change Management, Project Management or Environment [81].

RUP splits a project life cycle into four phases: Inception (scope the system to define business costs and budget), Elaboration (analysis of the problem domain and initial shaping of the project architecture), Construction (coding: components and features of the system are developed) and Transition ("transit" the system from development into production, ready for the end user, and check quality) [81].

11.4.4 Scrum

An iterative framework for project management

Scrum is an iterative, incremental framework for project management, a skeleton process that contains a set of roles and practices [82]. It aims to create self-organizing teams based on co-location of team members, verbal communication and discipline in the project.

One of its key principles is that customers can change their minds about what they want and need, so the problem cannot be fully understood and defined in advance: that's why it's important to have a team that is able to quickly respond to changes and emerging customer requirements.

The process relies on three pillars: Transparency (as much as possible), Inspection (as frequently as possible), and Adaptation (as quickly as possible).

What is reported in the following pages and more detailed information can be found in the Scrum guide, which can be downloaded for free at Scrum.org [73].

Scrum concept: roles, meetings and artifacts

Within the Scrum process there are three main roles:

- Product Owner, who represents the customer and other stakeholders
- Scrum Master, the project manager
- Scrum Team, the cross-functional group that does the analysis, design, implementation and test of the product.

Four types of meeting take place:

- Daily Scrum, which is held every day at the same time and place, is timeboxed (15 min), and involves the core roles listed above. During the meeting each team member is asked: "What did you do yesterday?", "What will you do today?", "What's on your way?" (i.e. "Which problems do you have?" – the Scrum Master is the one who should facilitate the resolution of these problems).

- Sprint Planning Meeting, which is held at the beginning of each Sprint (the 7-30 days iteration), involves the Product Owner and the Team, and will take no more than 8 hours. In this meeting the work to be done is selected and the Sprint Backlog (that is about the time and resources needed to do that work) is prepared.

- Sprint Review Meeting, which is held at the end of each Sprint and lasts not more than 4 hours. It reviews what has been done and what hasn't, and presents the completed work ("demo") to the stakeholders.

- Sprint Retrospective, which is held at the end of each Sprint, has a 3 hours time limit and is attended by the Team Members. They reflect on the past Sprint and they are asked "What should we keep doing?", "What should we stop doing?" and "What should we start doing?".

Three different artifacts are produced:

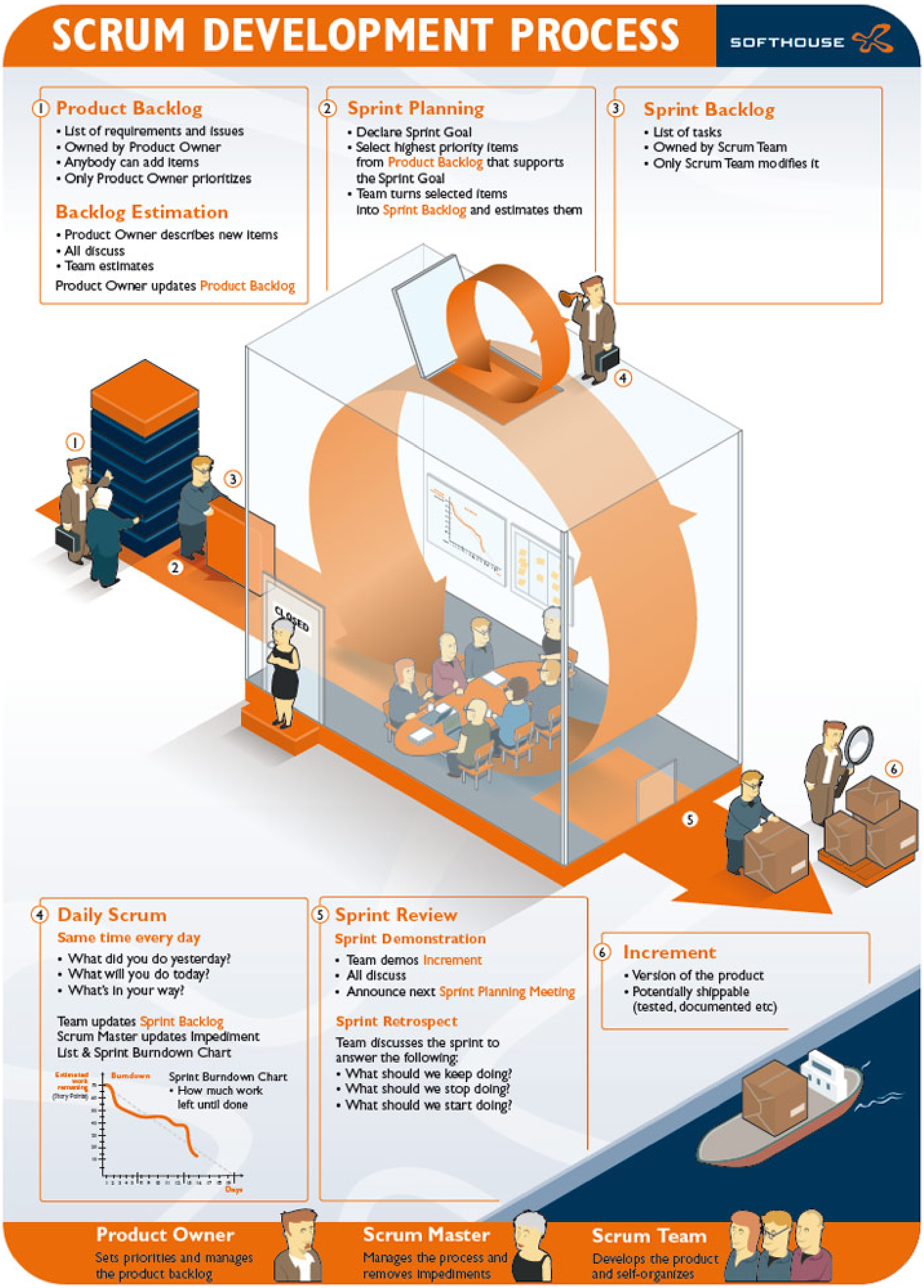
- Product Backlog, a list of descriptions of possible features sorted by importance.

- Sprint Backlog, a list of the work the team should do during the Sprint, where the features are broken down into tasks. A Task Board may accompany the Spring Backlog, describing the state of each task ("to do", "in progress", "done").

- Burn Down Chart, a publicly displayed chart that shows the daily Sprint progress.

All these elements as well as the logic of the Scrum process are depicted in the Scrum Development Process poster (Fig. 11.4).

Fig. 11.5 is a photo of how a Scrum room may look like. The picture has been taken at SAAB Training and Simulation in Huskvarna, Sweden.



SCRUM DEVELOPMENT PROCESS

SOFTHOUSE

1 Product Backlog

- List of requirements and issues
- Owned by Product Owner
- Anybody can add items
- Only Product Owner prioritizes

Backlog Estimation

- Product Owner describes new items
- All discuss
- Team estimates

Product Owner updates Product Backlog

2 Sprint Planning

- Declare Sprint Goal
- Select highest priority items from Product Backlog that supports the Sprint Goal
- Team turns selected items into Sprint Backlog and estimates them

3 Sprint Backlog

- List of tasks
- Owned by Scrum Team
- Only Scrum Team modifies it

4 Daily Scrum

Same time every day

- What did you do yesterday?
- What will you do today?
- What's in your way?

Team updates Sprint Backlog
Scrum Master updates Impediment
List & Sprint Burndown Chart

Sprint Burndown Chart

- How much work left until done

5 Sprint Review

Sprint Demonstration

- Team demos Increment
- All discuss
- Announce next Sprint Planning Meeting

Sprint Retrospect

Team discusses the sprint to answer the following:

- What should we keep doing?
- What should we stop doing?
- What should we start doing?

6 Increment

- Version of the product
- Potentially shippable (tested, documented etc)

Product Owner
Sets priorities and manages the product backlog

Scrum Master
Manages the process and removes impediments

Scrum Team
Develops the product and self-organizes

Fig. 11.4 Scrum Development Process poster (used at SAAB Training & Simulation)

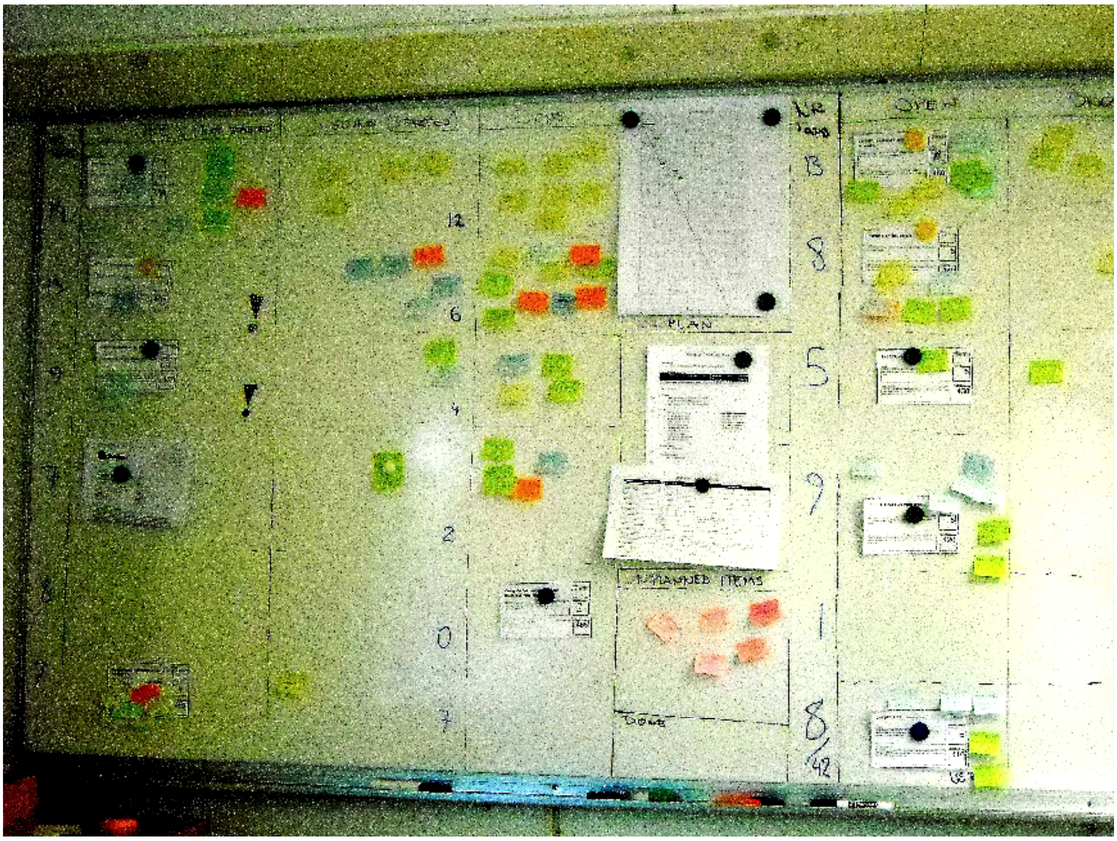


Fig. 11.5 Photo of a Scrum room (picture taken at SAAB Training and Simulation)

11.4.5 XP

A timeboxed software development methodology

Extreme programming, known as XP, is a timeboxed software development methodology to improve software quality and responsiveness to changing customer requirements. It involves programming in pairs, doing extensive code review and unit testing, adding features when actually needed, and frequent communication with the customer.

The goal is to organize people to produce higher quality software more productively and to reduce the cost of changes in requirements by having multiple short development cycles [77].

XP concept

Four basic activities are performed within the software development process: Coding (without code there is no working product), Testing (Unit tests, to check that each feature works as intended, and Acceptance tests, to check that requirements are met - the more tests, the better), Listening (customer needs must be well understood) and Designing (the logic of the system needs to be explained) [77]. For each of these activities, best practices are provided.

Five values underlie the XP concept: Communication, Simplicity (do not do more than what is actually needed), Feedback (from the system, the customer and the team), Courage (review the system so that future changes can be implemented easily, remove obsolete or useless code, be persistent in solving problems) and Respect (strive for quality, seek for the best design, do not delay the work of others or make tests fail) [77]. Moreover, change should be embraced.