UNIVERSIDAD NACIONAL DE COLOMBIA

Knowledge transfer measurement methodology for Software Requirements, a case study

José Jairo Camacho Vargas

Universidad Nacional de Colombia

Faculty of Engineering, Systems and Computer Engineering

Bogotá D.C, Colombia

2015

Knowledge Transfer measurement methodology for Software Requirements, a Case study

José Jairo Camacho Vargas

Thesis submitted as a partial requirement to obtain the degree of:

Master in Computer and Systems Engineering

Assessor:

Ph.D. Marcela Sánchez-Torres

Research lane:

Knowledge management and software engineering

Research group:

Griego

Universidad Nacional de Colombia

Faculty of Engineering, Department of Industrial and Systems Engineering

Bogotá D.C, Colombia

2015

(Motto)

The energy of the mind is the essence of life.

Aristotle

Acknowledgments

I want to express my gratefulness to: my God, family and assessor, for their support and patience.

Abstract

The purpose of this work is to present a proposal methodology for knowledge transfer measurement in software requirements. To obtain results, a methodology composed of four stages was defined: i) review of the knowledge transfer background in software engineering, in order to identify existing efforts in knowledge transfer measurement, ii) characterization of the software requirements process from the knowledge transfer point of view, thus, finding common factors regarding variables and indicators suitable for measuring purposes, iii) define a proposal methodology based on variables and indicators found, data gathering methods, statistical tools and helping documentation, iv) testing the proposal in order to provide feedback, using a case study.
Principal results are: seven groups of factors mapping software requirements process stages against knowledge transfer steps, resulting in 115 indicators and 24 variables; 2 variables definition for knowledge transfer initialization stage and software requirements elicitation step mapping, which didn't had any variable or indicator. Likewise, it was identified that exists a correlation between knowledge transfer and software requirements, the better knowledge transfer the better software requirements.

Furthermore, the feed back gathered indicates that motivation variable defined is the more influential variable in the software requirements process according 41.67% of respondents, over other variables as: abstraction, methodology and time access availability, each one with 16.67% of respondents, and understandability with 8.33% of respondents.

Last, this work allows analyzing the influence of knowledge transfer indicators in software requirements quality attributes.

Key words: Knowledge management, knowledge transfer, software engineering, software requirements, software requirements metrics, knowledge transfer measurement.

Resumen

El propósito de este trabajo es presentar una propuesta metodológica para la medición de transferencia de conocimiento en los requisitos de software. Para obtener los resultados, una metodología compuesta de cuatro pasos fue definida: i) revisión de las bases teóricas de transferencia de conocimiento en ingeniería, para identificar esfuerzos existentes en medición de transferencia de conocimiento, ii) caracterización del proceso de requisitos de software desde el punto de vista de la transferencia de conocimiento, y de esta manera, encontrar factores comunes con respecto a variables e indicadores adecuados para los propósitos de medición, iii) definición de una propuesta metodológica con las variables e indicadores encontrados, métodos de captura de datos, herramientas estadísticas y documentación de ayuda, iv) prueba de la propuesta metodológica para proveer una retroalimentación, usando un estudio de caso.

Los resultados principales son: siete grupos de factores mapeando las etapas del proceso de requisitos de software contra los pasos de transferencia de conocimiento, resultado en 115 indicadores y 24 variables; 2 variables definidas para el mapeo entre la etapa de inicialización en transferencia de conocimiento y la etapa de elicitación de requisitos de software, el cual no tenía ninguna variable o indicador definidos. Igualmente, fue identificada una correlación entre transferencia de conocimiento y requisitos de software, a mejor transferencia de conocimiento mejores requisitos de software.

Además, la retroalimentación obtenida indica que la variable de motivación definida es la más influyente en el proceso de requisitos de software según el 41.67% de los encuestados, por encima de otras variables como: abstracción, metodología y disponibilidad de tiempo, cada una con 16.67% de los encuestados, y comprensibilidad con el 8.33% de los encuestados.

Por último, este trabajo permite analizar la influencia de los indicadores de transferencia de conocimiento en los atributos de calidad de requisitos de software.

Palabras clave: gestión de conocimiento, transferencia de conocimiento, ingeniería de software, requisitos de software, métricas de requisitos de software, medición de transferencia de conocimiento.

Content

Figure list

Table list

Abbreviation list

Abbreviations

Abbreviation	Term
KT	Knowledge transfer
SE	Software engineering
SR	Software requirements
RE	Requirements engineering
KM	Knowledge management

Introduction

Software Engineering has been recognized as a knowledge intensive application discipline (Rus & Lindvall, 2002), (Dingsøyr, Bjørnson, & Shull, 2009) and (Ward & Aurum, 2004). For this reason, in the last decade there has been an increasing interest about knowledge management in software engineering. In particular, the processes of knowledge codification and knowledge sharing have received most attention and they have been researched in diverse ways.

Other authors have argued about the relevance of knowledge transfer processes in knowledge management (Albino, Garavelli, & Gorgoglione, 2004; Kumar & Ganesh, 2009) and the importance of knowledge transfer in software engineering (Argote & Ingram, 2000; Inkpen & K. Tsang, 2005).

In this sense, there is a consensus about the importance of the knowledge transfer process, however, there is still a debate among what the knowledge transfer process really is, because knowledge is not only tangible and linked to cognitive processes inside people's brains but also is said to be particular for everyone (Krogh, Nonaka, & Aben, 2001; Nonaka & Toyama, 2003), therefore, is not easy to measure.

Measure the KT is a problem, because there not exist any clear model that allow a quantitative and/or qualitative approximation to KT, specially due to the KT is not the only way to create new knowledge (Awad, 2005; Kumar & Ganesh, 2009).

The phase of collection and specification of requirements play an important role because is where the business needs are translated to technical language and is set the scope of the software project (Hagge L., 2005; Pilat & Kaindl, 2011) . The problem of carries the business concepts across all the software development steps has not been studied and

represents and interesting percent of fail causes (23% StandishGroup) in the software development projects.

Even if the software chaos report from the StandishGroup has been criticized (Jørgensen & Moløkken-Østvold, 2006) (Holtsnider, Wheeler, Stragand, & Gee, 2010), and the 189% overrun percentage in software projects has been reduced to 34% and 33%, the 68% of projects that still fail are because of the poor requirements specifications.

The scope of this thesis is to get an insight in SR as a KT process, limited to a study case. The sample used is not statistically representative, since the scope is to understand how KT for SR happens and how it could be measured. The purpose of the present work is to design a proposal methodology to measure the knowledge transferring process of software requirements (SR) in a software development organization. Understanding a proposal methodology as a collection of procedures, techniques, tools and help documentation (Avison & Fitzgerald, 2006). Specifically, this research concentrates on:

1. Establish what we know about KT in software engineering and factors who affect it. This work defines a general background about KT in software engineering.

2. Characterize the KT process in the SR process. This work maps the KT activities with SR processes activities.

3. Design a methodology that involves metrics for the KT process in the SR process based on their characterization. This work develops a series of indicators for KT measure.

4. Test the proposed methodology in a software project. This work consists of apply the methodology created using a study case, to validate it.

The methodological process used in this research was organized in four stages as follows:

Stage 1: Review of the knowledge transfer background in software engineering.
In this stage different approaches of KT in software engineering were identified. The goal in here was to identify existing efforts in KT measurement, thus, recognizing enabler factors for KT and get an insight in how KT measurement have been done until now. The review was done using the systematic literature guidelines given by Kitchenham (2007), there were no constraints on papers data, all papers until 2012 were consulted.

Stage 2: Characterization of the software requirements process from the knowledge transfer point of view.
Since this thesis involves the measurement of KT, a post positivism point of view is used, so, KT in SR is determined and reduced, finding common factors regarding variables and indicators suitable for measuring purposes. Another review was done, since initial review result in no suitable approach for KT measurement. In other words, nor variables or indicators were found. The second review was focused in identify the stages of KT process and how SR process steps match in such KT stages. So, a mapping strategy was defined to compare KT process stages against SR process steps.

Due to this thesis is focused in KT, the second review doesn't focus in SR, SWEBOK v3 was chosen as the guide to SR because it describes generally accepted knowledge about software engineering. Its 15 knowledge areas summarize basic concepts and include a reference list pointing to more detailed information. For SWEBOK Guide V3, SWEBOK editors received and replied to comments from approximately 150 reviewers in 33 countries, it also gained international recognition as ISO Technical Report 19759.

Once the mapping was done, for each KT-SR matched stage a group of factors was identified. And from those factors, variables and indicators was identified following a dimension analysis and measurement aspects, all of this grouped by referents.

Stage 3: Proposal methodology definition based on variables and indicators found, data gathering methods, statistical tools and helping documentation.

From previous stage, an initial indicators framework was defined. Due to that KT in SR is not widely studied in literature found, this research is also explorative in order to understand how KT happens in SR, thus, an interview is proposed to validate if the full initial framework.

Once validated, indicators are used in a questionnaire, which was used as a data collection strategy to make an empiric observation and measurement. Because of the questionnaire, it is need to provide validation for internal consistency, so Cronbach's alpha is proposed since its widely used in the literature review done in previous steps.

Likert question types were defined for the questionnaire, and also a numeric range mapping is provided to quantify answers. Four responses levels are provided, each one with a numeric value.

Then, statistical analysis is proposed, taking advantage of numeric answers. Pearson's index is proposed as a correlational tool.

Stage 4: Testing the proposal in order to provide feedback, using a case study.

First of all, a kickoff meeting was done for instrument refinement, then a case study is conducted to get feedback from the proposal methodology, so, the instrument defined in the previous stage is implemented. Data was gathered using an online questionnaire, then data was analyzed as specified by the proposal methodology, and documentation was made.

Figure 1 shows the methodology process followed in this work.

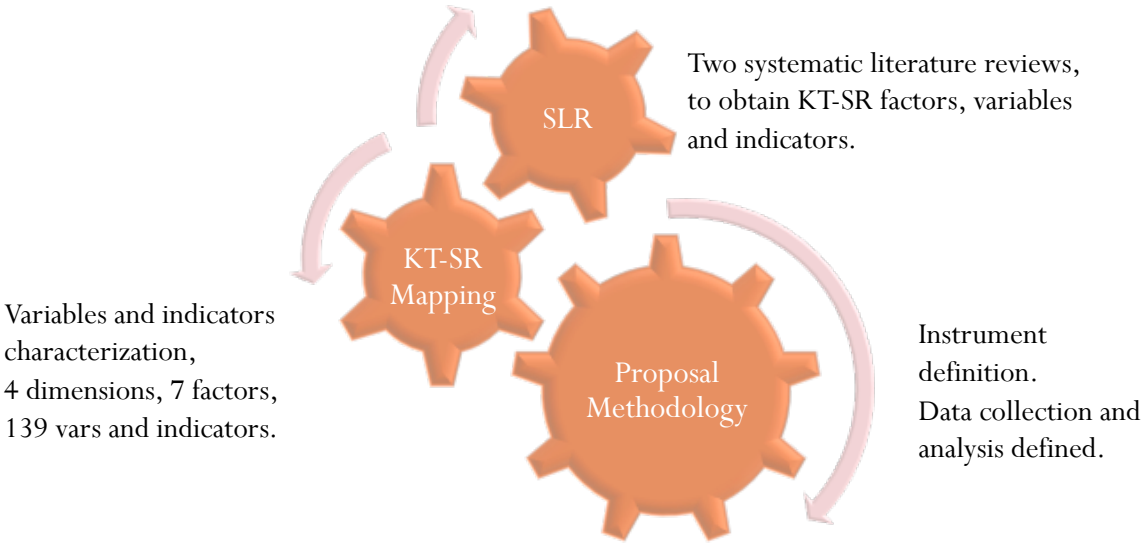


Figure 1. Methodology process.
Source: Self-elaboration based on process methodology.

Achievements: 1) a systematic literature review done about KT and SE, 26 referents were found about KT measurement, additionally 23 referents about KT in SE, but not one specify a clear model or indicator for KT measurement, just 4 referents make any emphasis in SR; 2) 4 KT stages was defined after a systematic literature review, then, was mapped against the 4 SR steps taken from SWEBOK v3. Based on such mapping, 7 groups of factors who affect KT in SR. Those factors were built based on 14 referents and after the mapping 115 indicators and 24 variables was found; 3) Proposal methodology was depicted, 16 variables covering SR quality and KT were defined as the initial framework questionnaire, since KT and SR can change depending the organizations, a kickoff meeting was added to the proposal methodology in order to refine the initial framework and adjust it to the organization reality. Questionnaire internal consistency and correlation index was defined as the statistical tools to be applied, for data collection and data analysis. Feedback for the proposal methodology was encouraged; 4) A study case was conducted in a Brazilian company. Based on the kickoff meeting, it was decided to use the full initial framework. Data was gathered using Google Forms, Cronbach's Aplha and Pearson's Index was used for questionnaire internal consistency and for variables correlation. It was found a strong correlation between KT and SR quality, likewise motivation was the principal enabler for KT.

Chapter one depicts the background about KT in SE as the result of a systematic literature review, answering objective 1. Next, chapter 2 focuses in KT for SR, there SR process is view as a KT process, reaching objective 2. Then, chapter 3 shows the proposal methodology for KT measurement in SR, fulfilling objective 3. Afterwards, chapter 4 presents the case study, in order to test the methodology, meeting objective 4. Later, in chapter 5 appears the conclusion of this work. At the end, Appendixes with the full questionnaire answers and calculations are included.

1. Knowledge Transfer in Software Engineering.

There is a review of different concepts about KT applied to software engineering. The review was made based on a systematic literature review. We start with a background about knowledge transfer in section (1.1). Next in section (1.2) what we know about knowledge transfer in software engineering. And finally, in section (1.3) the conclusions of this chapter are presented.

1.1 Knowledge Transfer

On the one hand, knowledge has been defined as the information and experience grouped usefully in some context (Alavi & E. Leidner, 2001), and literature shows a consensus about the taxonomy that represents knowledge as tacit and explicit (Krogh et al., 2001)(Nonaka, 2007). On the other hand, transfer means to pass an element form one side to other (Watson & Hewett, 2006; Borgatti & Cross, 2003). In other words, KT means to pass useful information and experience from one context (project) to other place (inside or outside of an organization).

Nevertheless, such transfer, according to some authors, cannot be done (Krogh, 2003) due to the fact that knowledge is personal and unique. Every time knowledge passes from tacit to explicit (Garavelli, Gorgoglione, & Scozzi, 2002), new knowledge is generated so it is different from the previous one. In this way, the exactly transfer of knowledge cannot be possible.

It should be noted that KT is different from knowledge sharing (Argote & Ingram, 2000; Kumar & Ganesh, 2009) since the fact that one person shares knowledge does not mean that he/she already did a transfer. Consequently, entity A (person, business unit or company) transfers knowledge to entity B, just when B is able to apply it in a useful way in

his own context. By the same token, it can be said that only sharing knowledge has occurred.

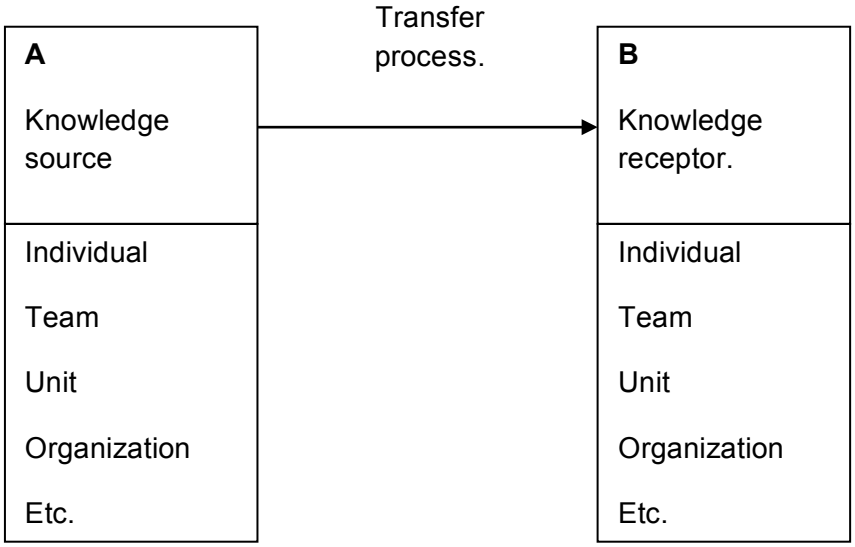


Figure 2. Knowledge transfer basic model.
Adapted from Kumar & Ganesh (2009)

Knowledge sharing is important as a KT enabler, even sharing alone is not enough to make the transfer occur. This is very important because, until now, the greatest advances in knowledge management applied to software engineering have been done at this level – share- such a process is known as knowledge codification (Farenhorst & Vliet, 2009; Garavelli et al., 2002; Gosain, 2007).

KT is more than mere codification because it demands more than building “knowledge” bases (data and information) (Kumar & Ganesh, 2009). Those bases ended being just data repositories – and thus, them can only code the knowledge. Knowledge is related to a human process and it could only be generated through cognitive process inside people’s minds (Carayannis, 1999).

1.2 Knowledge Transfer in Software Engineering

Due to the intensity of the use of knowledge in software engineering (Tesch, Sobol, Klein, & Jiang, 2009; Wilkesmann & Wilkesmann, 2011), the software engineering processes are at an interesting place to the KT. A general view of a software development project

could have five phases (requirements, design, development and testing, integration and maintenance) with the classic cascade model (Karlsen, Hagman, & Pedersen, 2011).

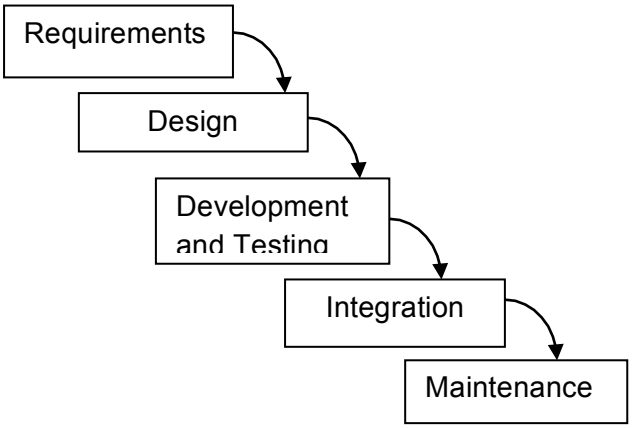


Figure 3. Cascade model.
Adapted from Karlsen et al. (2011)

Although the cascade model could look like obsolete nowadays, certain authors (Karlsen et al., 2011) continue referencing it because other models have appeared based on this model, those models include some improvements, for instance, iterations, recursion or parallel steps, but in the end, the basic phases of the software developing cycle are the same. Such phases could be executed in different order or with a different focus.

Along the requirements phase, there is an interaction between the technical team and the costumers, which are the owners of the business knowledge. These costumers must transfer the aforesaid business knowledge to the analysts, so that they can design models that should help to transfer business knowledge to technical knowledge in models that describe the software such as class diagrams, components, and so on (Havlice, Kunstar, Adamuscinova, & Plocica, 2009; Ward & Aurum, 2004).

Likewise, within the technical team, KT occurs in software development (techniques and programming procedures) and software testing (test cases, scenarios, etc.).

1.2.1 Knowledge transfer within software development multinationals

In the literature review some articles which treat the topic of KT in multinationals were found out; they all claim that very few research has been done on the topic in multinational environments, where, there is not only a distance issue, but also, cultural facts (Ambos & Ambos, 2009; Duan, Nie, & Coakes, 2010; Niederman, 2005).

Those authors do a list of possible factors that affect transfer in such environments, being the cultural factor the most troublesome. To avoid such difficulties, they define a mechanism to code the knowledge, for instance, internationalization tools to mitigate the idiomatic differences are made by coding and some others generate more sophisticated mechanisms as ontologies to define a common language.

With those codification tools, it is possible to facilitate the KT, because the physical and cultural gaps can be reduced.

In brief, the topic of KT in multinational environments addresses the issue of information and experience transfer of successful projects from one organizational unit to other.

To carry out what was said before, authors state diverse factors and hypothesis that are supposed to increase the transfer effectiveness. The principal role of technology in this aspect is to serve as an information repository with relevant information about the work to be done, as well as a collaboration tool to mitigate distance between people (Ambos & Ambos, 2009; Aurum A., 2008; Y.-J. Chen, Chen, & Chu, 2008; Duan et al., 2010).

Figure 4 shows a set of common factors taken from the review. These factors expose characteristics that should be considered for an effective KT. It just shows factors at organizational levels, this means only elements that inside the organization administration can affect the KT effectiveness. Promptly in the review, three groups of factors or dimensions were found out: structural, relational and cognitive.

Then, the structural dimension deals with how the communication inside the organization is done; this communication can be formal or informal, depending on the communication channels that are used. For instance, one type of formal communication can be meetings

or memorandums, by contrast, an informal way could be a socialization during a coffee break or the use of social networks.

The second dimension is relational dimension that deals with factors of people and their culture inside the organization; such factors include the confidence between people, bosses and subordinates, the organizational commitment of these people, the remuneration capability of the organization, and the factor of the identity towards the organization and the work done.

The last dimension, the cognitive dimension, references the capability of the organization management to articulate strategies along all of the processes and people, together with a factor of organizational culture to provide the dimension mentioned above.

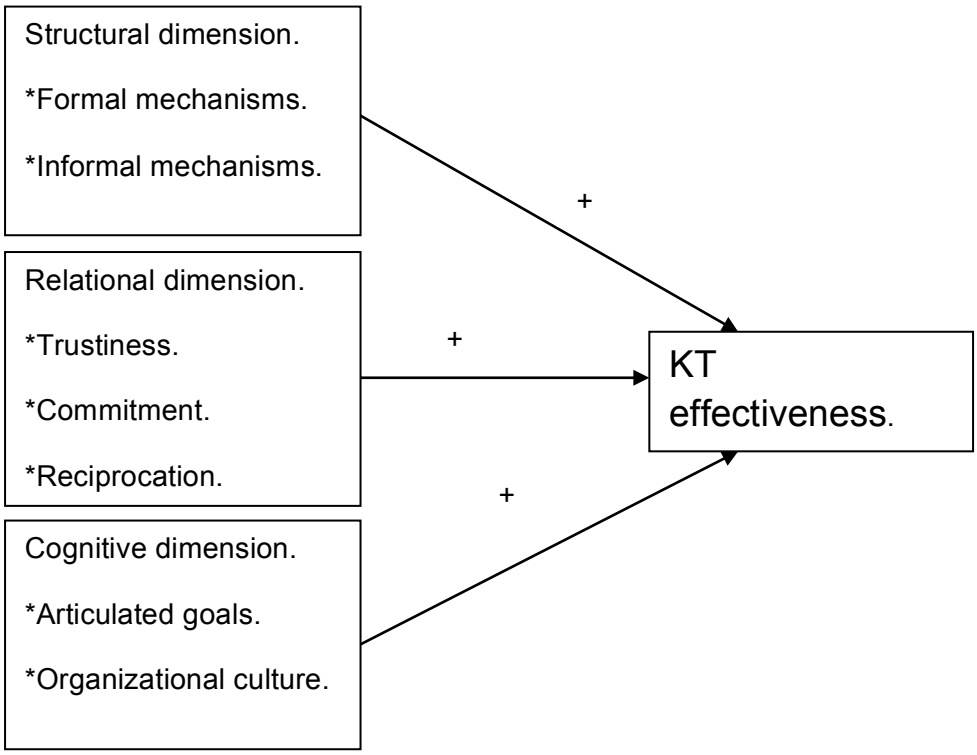


Figure 4. Knowledge transfer factors in multinational environments (1)
Based on (J.-S. Chen & Lovvorn, 2011).

Meanwhile, Figure 5 shows the coordination elements that influence the KT. The objective is to see that it is not only necessary to take into account the organizational factors

mentioned above, but also the technology that supports the KT processes, especially, the ones related to mitigating extenuating factors.

Such technology tools can be knowledge bases: repositories in relational or document oriented databases, with useful information about the business processes; collaborative tools: such as social networks and, in general, any technology that supports people communication and the building of collective concepts, such as blogs or forums (Ambos & Ambos, 2009; Eric Ras, Gabriela, Patrick, & Stephan, 2005).

Technology tools help to mitigate physical distance, because the internet/intranet does not require that the work teams stay together in the same place. In addition, translation tools and ontologies can mitigate ambiguity in the texts and idiomatic differences (Kjærgaard, Nielsen, & Kautz, 2010). More importantly, there are face-to-face coordination methodologies (face to face meetings, for instance at the same location), which are traditional but are more sensitive to the effects of geographic and linguistic distance (J.-S. Chen & Lovvorn, 2011).

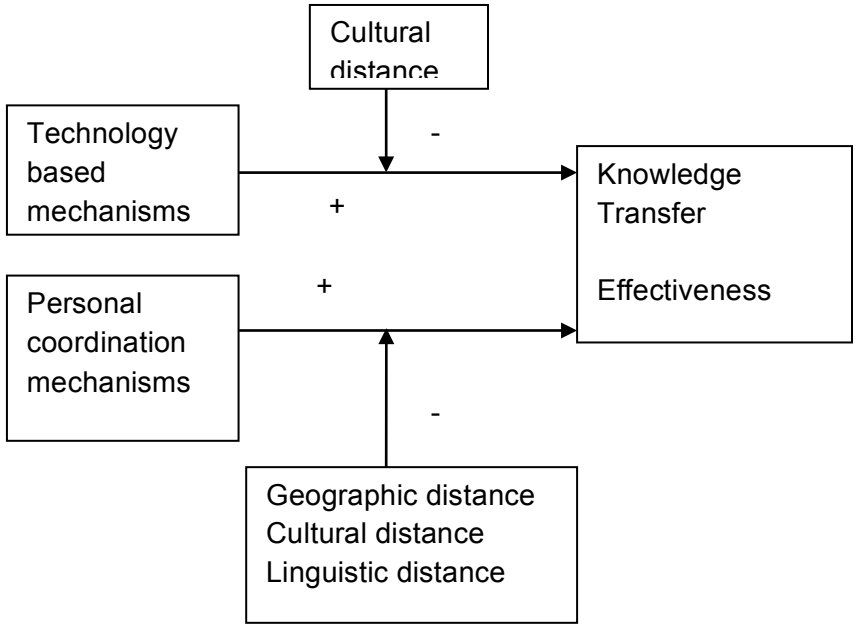


Figure 5. Knowledge transfer factors in multinational environments (2).
Based on (Ambos & Ambos, 2009)

1.2.2 Knowledge Transfer in Agile Models

KT is evidenced by feedback between people. The agile models promise to decrease documentation in favor of coding speed, leaving the knowledge inside the people's head, however, it favors the knowledge flows while making periodic meetings (Karlsen et al., 2011). Basically, one might think that an agile environment is more adequate to the KT than a traditional one (cascade, RUP), in which a series of pre-requisites are demanded in order to advance to a posterior step.

Agile environments facilitate to share the knowledge because the teams work on iterations that allow a continuous feedback, not only inside the technical team, but also with clients and owners of business knowledge (Larman & Basili, 2003; Whitworth, 2006)

Regardless of the agile methodology, the goal is to favor the interactions between people instead of processes and tools, the software work instead of detailed documentation, the collaboration with clients instead of contract dealing and with this, to respond to the variant requirements as it progresses.

There are various agile models, and an abstraction of them could be seen as an iterative cascade model (due to the software development phases) iterative with multiple interactions. Although there is an iterative software model, the difference with an agile method is that iterations are done between little, very specific requirements/functionalities, and the traditional documentation is minimal.

Figure 6 shows such interactions, in which KT is also performed, since it does not just involve a relation among developers, software testers and analysts, but includes the owner of the business and the expert in the process to be automated in the KT (Ambos & Ambos, 2009; Koskinen, Pihlanto, & Vanharanta, 2003; Marshall & Brady, 2001).

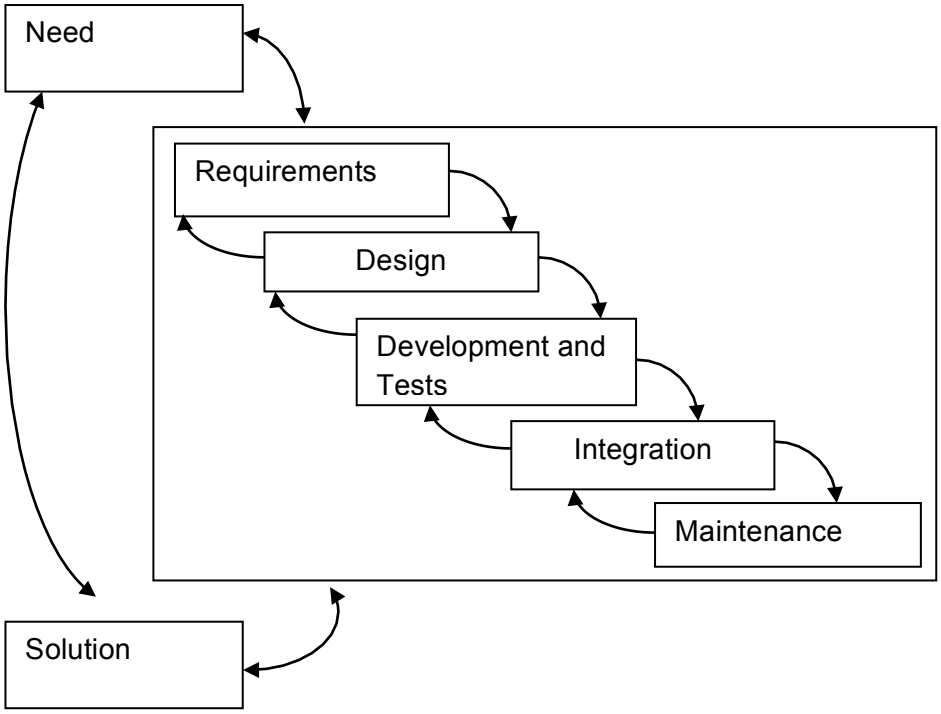


Figure 6. Agile model generalization.
Based on (Awad, 2005)

1.2.3 Knowledge transfer among the projects

An interesting point of view for the KT is that it can take place in an isolated environment inside the organization, which is important because it facilitates the processes given that the team is inside the same organization. In contrast, the transfer between organizations could be difficult due to intellectual protection issues, which prevent the flow of knowledge (Awad, 2005; S. M. . Jasimuddin, Connell, & Klein, 2012; S. M. Jasimuddin, 2007).

Inside the same non-multinational organization, the concern is to achieve a link between culture, processes and their supporting technology to facilitate the KT (Lee & Shiva, 2009).

Figure 7 shows the current methods used for the KT. There are various groups of people that represent the projects that could have or not experts. These experts form an expert net, which could be managed in a formal way through directories, where experts could be

contacted for a topic, or could be undocumented nets, as friend or colleague nets. Ideally, such nets should be documented in a directory or software that leads to the experts.

Likewise in Figure 7 appears a knowledge base with an ontology that defines the business process language. Such knowledge bases are useful to keep the history of the management done in the projects, and their goal is to preserve the best practices or key factors that have contributed to the success of the projects.

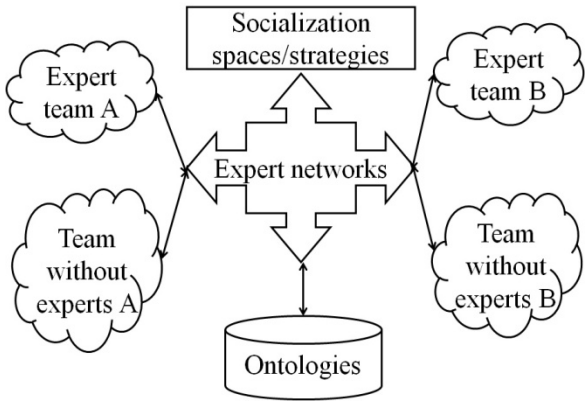


Figure 7. Experts net with a knowledge base.
Based on Schneider (2009)

The concern here is to reproduce the knowledge by taking advantage of the experts in an organization or business unit, so the others could be benefited of their experience. It is not just about repositories with knowledge bases or ontologies that represent the domain of the problem, but the technology tools that must lead to the collaboration and access to the experts so that they could be reached through their documents and face to face, improving the KT (Havlice et al., 2009). This has not had a big development according to the review done, so it could be a good topic to research.

1.2.4 Knowledge transfer between people

Finally, in a more atomic level, the KT between two people is studied. The SECI model (Nonaka & Toyama, 2003), defines a series of steps that are followed in the learning process of a person. Which are: a) socialization, where a person A socializes his/her

experience and knowledge with other person B; b) exteriorization occurs in person B when he/she can define concepts in their own context about the knowledge acquired; c) person B does combination when applying his/her new knowledge and builds prototypes, finally, d) the knowledge is internalized in person B through practice, so the knowledge becomes a part of his/her mental models, believes, abilities, etc. Into the previous general model of knowledge management, KT could be seen in the existence of two persons; apprentice and master, where the apprentice in turn can be an expert in certain topics and a master can be an apprentice in others (Wilkesmann & Wilkesmann, 2011). This shows that in general any expert person may also have the need of learning and acquiring new capabilities for different projects, in this case about software.

The goal of transfer between people is that they meet generally in an informal way, to treat the issues of the organization and give each other pieces of advice on how to carry out the work in the best way according to their experience. Like this, social methodologies were born, as the coffee breaks, where people get involved to share their experience.

Figure 8 shows this kind of interaction, where the objective is not a plain transfer, because of the personal nature of the knowledge, and instead of doing a transfer, what is done is building new useful knowledge in one or more contexts(Krogh et al., 2001).

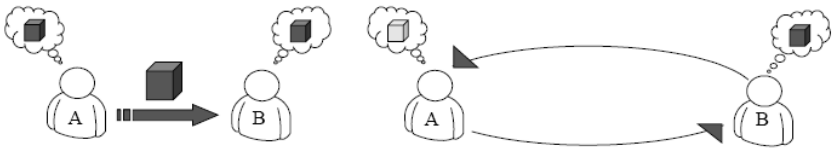


Figure 8. Knowledge transfer between people model.
Taken from Wilkesmann & Wilkesmann (2011).

1.2.5 Factors helping knowledge transfer

Majority of the research has been done using surveys trying to determinate the effectiveness of the methodologies applied for the KT.

Surveys are used to be created from a proposed model and are validated sending it to different companies. Ideally some authors propose consistence indexes to measure the gathered responses and the integrity of the questions (Wilkesmann & Wilkesmann, 2011).

The factors to measure always come from hypothesis about people behavior, is used to presume that for the KT occurs, a predisposition of the people to share knowledge should exist (Karlsen et al., 2011).

Some authors claim that people interviewed usually are willing to share their knowledge and emphasize the importance of knowledge sharing to carry out their activities (Y.-J. Chen et al., 2008; Liebowitz & Suen, 2000).

It could be interesting to measure the previous purpose in the local industry because the reviewed surveys recommend reapplying them in local environments to try to verify and validate their questions and answers (Aurum A., 2008).

Typically, factors have two components, one organizational and one technologic. The organizational is used to measure the people and business unit's readiness towards knowledge management, and especially towards KT. The technology is used to see how far could exist collaboration tools and knowledge bases to support the KT process.

Figure 9 depicts the importance of organizational factors joined with a set of technology to facilitate the KT. There it appears a social space (coffee) and socio technical nets, because is natural that people relate between each other. Besides there are communication tools and experience repositories where finally an agent appears on the middle in order to help the articulation between the members.

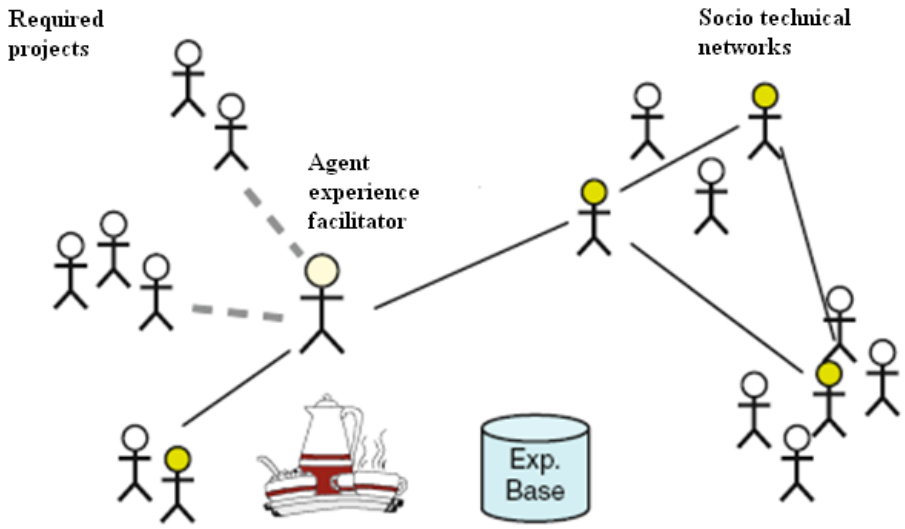


Figure 9. Knowledge transfer mixed techniques. Experience base and people networks. Taken from (Schneider, 2009).

1.2.6 Knowledge transfer measurement

There are few proposed metrics to measure KT. It is argued that KT is something difficult to measure, because it cannot be measured directly, and always is measured indirectly. If it is tried to be measured by the knowledge created, it could be a mistake due to the KT is not the only existing way to create new knowledge (Karlsen et al., 2011; Wilkesmann & Wilkesmann, 2011). To deal with this, is used to ask for the KT perception in the organizations and people.

Table 1 summarizes the efforts found out about KT measurement, unfortunately there aren't clear indicators or metrics, nor any mean of how to measure. The most common metrics are those who deal more with final elements more than the mere KT. For example, there are some authors who emphasize more in final products, innovation quantity or new concepts generated, however, it is not only with the achievement of final products or the generation of new concepts that it could be said that the KT is successful (Abdullah, Selamat, Cob, & Sazaly, 2011; Chen C.-J.a Shih, 2009; Gardner, Fong, & Huang, 2010; Liebowitz & Suen, 2000; Mei, Wang, & Cao, 2011).

Furthermore there may be more interesting metrics for the organization, for instance oriented metrics to the gaining of money or value generated, such as earnings that come after a project that had KT or even the return of investment over a knowledge active, e.g. a patent (Crowne, 2009; Gardoni, Frank, & Vernadat, 2005; Gorschek & Davis, 2008; Li J.a Moe, 2010; Liebowitz & Suen, 2000; Rezgui, Hopfe, & Vorakulpipat, 2010).

Either the commercial value could be taken into account, for instance the gain of a new market, the client's satisfaction, gained clients, increased sales by clients etc. Another metric could be the value created by research and development activities and the return of invest by each trained employee (KT) (Carayannis, 1999; Desmarais et al., 2009; Formentini & Romano, 2011; Salger, Sauer, Engels, & Baumann, 2010; Szulanski, 1996).

Some metrics more oriented to KT, are focused on the quantity of knowledge (in knowledge bases) frequently acceded or reused. And measure how many people have shared their knowledge (Chakraborty, Sarker, & Sarker, 2010; Duan et al., 2010; Hagge L., 2005; Ko, Kirsch, & King, 2005; Kyaw P., 2003; Poort E.R.a Pramono, 2009).

In short KT cannot be measured directly, but it is measured in a qualitative way using organizational and technological factors, together with final products obtained from such transfer (Chau, Maurer, & Melnik, 2003; Liebowitz & Suen, 2000).

It should be noted that the measurements are done by surveys from which the questions are related to the factors to measure, for instance, if it is desired to know if the transfer process X was effective, it is asked if the people consider that they could learn from process X and in turn if it was useful to be applied to certain project Y. To this some authors (Chen C.-J.a Shih, 2009; Karlsen et al., 2011; Liebowitz & Suen, 2000; Wilkesmann & Wilkesmann, 2011) show examples of questions, but in general those kind of articles do not show the full questionnaire used in the organizations.

Table 1. Effort summary on KT measurement.

Author	Knowledge	Indicator	Tool / Method
Karlsen et al. (2011), Wilkesmann & Wilkesmann (2011).	Creation, sharing	Knowledge created	Questionnaire
Abdullah et al. (2011), Chen C.-J.a Shih (2009), Gardner et al. (2010), Liebowitz & Suen (2000) and Mei et al. (2011).	Creation	New products and concepts created	Not specific
Crowne (2009), Gardoni et al. (2005), Gorschek & Davis (2008), Li J.a Moe (2010), Liebowitz & Suen (2000) and Rezgui et al. (2010).	Creation	New patents, return on investment, money, income	Questionnaire
Carayannis (1999), Desmarais et al. (2009), Formentini & Romano (2011), Salger et al. (2010) and Szulanski (1996).	Creation, Share	Market share, money, income, ROI per trained people	Questionnaire
Chakraborty et al. (2010), Duan et al. (2010), Hagge L. (2005), Ko et al. (2005), Kyaw P. (2003) and Poort E.R.a Pramono (2009).	Codification, share	FAQ access count, Technology usage,	Questionnaire
Chau et al. (2003) and Liebowitz & Suen (2000).	Sharing	Technology usage	Questionnaire

Table made by authors based in the literature review.

Source: self-elaboration based in the literature review done.

1.2.7 Knowledge transferring measurement in SE

Even when the KT is difficult to measure, talking about KT measurement in SE is not new. Maybe there are not clear ways or at less, a clear path to measure KT, some authors have done an effort to make an approximation to SE measurement from the KT viewpoint.

Table 2 synthesizes authors found out works about KT measurement in SE in 3 groups. First two are about what to measure, while the third group is about KT measure in SR.

Most of authors haven't choice one specific software process but instead of that, try to explain how KT happens in the software processes. Those efforts could be seen as authors pursuing KT measure through social capital variables like Bjørnson F.O.a Dingsøyr (2005); Boden, Avram, Bannon, & Wulf (2009); Wang & Yang (2008); Windiarti, Ferris, & Berryman (2011); Zhang Q. (2011) who state that trustiness, communication, respect and commitment are fundamental in the culture of an organization for people to share their knowledge and feel comfortable in doing it.

However, culture per se is not enough for the KT to succeed so other variables come into play, Baruch Y.a Lin, n.d.; Hongmei & Huidong (2008); Poort E.R.a Pramono (2009); Wah C.Y. (2005) states that incentives are important for people to share their knowledge, especially because people who have are highly competitive are the ones who have more knowledge and often such kind of people behaves apathetic to the knowledge sharing, so the coo-petition strategy is proposed.

In addition, S. M. Jasimuddin (2007) and Ren (2009) claim that communications style (virtual or face to face) are fundamental for the correct knowledge flow from experts to apprentices. Finally, among the first set is also the notion of intellectual capital as a mean of KT measure (C.-J. . Chen, Shih, & Yang, 2009).

As a second group of indicators there are the authors who focused their efforts in measure KT through technology tools and methods for KT, Conradi & Dybå (2001); Kuk (2006); Shou & Sun (2010) explain how communication being supported for the

technology can help knowledge flow, especially if there are an idiomatic or background difference between people.

Also, there are some methodologies like the communities of practice (Mestad A.a Myrdal, 2007) who ease the KT because of the enhanced communication (Qu G.a Ji, 2011; Zhang, Bao, Gao, & Guo, 2008) .

Finally, for the purpose of this study, in the third group appear six authors who are the ones, at the time of the review done for this thesis, who have study KT in SR. Eric Ras et al. (2005) see the KT transfer in SR as a matter of documentation, so proposes a weblog to store requirements information thus making easy to access the them, also make an addition for frequently asked questions over the requirements. This way, the metrics defined are the quantity of access to the weblog and FAQ´s viewed, resolved and cited.

Salger & Engels (2010) propose requirements specification through KT, this is done doing a double verification in a testing oriented software development, where there is an initial verification done by the stakeholders and a second verification done by an external development team, so the mean of measure is verification.

This could be sound quite strange, because according SWEBOOK client review of requirements is called verification while development team reviews are called validation, and those are two different processes. Xiaohong Shan, Jiang, & Huang (2010) made a model of KT for SR, but it does not make an effort to state a sort of metric, instead of that states that tacit knowledge is the most difficult to elicit and that physical and cultural distance affect the KT, also classifies the kind of knowledge according the knowledge holder.

Chau et al. (2003) addresses the issue of using agile methods as a KT enables, because of agile methods favor the communication thus making easy the tacit knowledge flow, intends to measure agility and quantity of communication.

Damian, Marczak, & Kwan (2007); Hagge L. (2005) state the importance of communication and define a kind of patterns to ease the SR process and KT through it,

what intends to measure is the SR specification, in other words, knowledge codification and their validation.

Finally, Pilat & Kaindl (2011) are the ones who bring a full view of SR from knowledge management point of view, they propose a methods that is tested on a study case. The method consist of identify knowledge holders and map the requirements versus the knowledge holders thus making clear who are the experts, such experts are the responsible for validate the requirements. From this model, some indicators could be measured such as: the perceived pay off (an incentive given to experts as a retribution for their knowledge) and efficacy (showing experts the importance of their knowledge and contribution for the organization); common vision, it means the quantity of people who agree that the project is important and have the willingness to participate in it, so, a collaborative environment is set, and the group identity is defined as a factor affecting KT for SR.

Table 2. KT measurement effort in SE.

Authors	Software process	What intends to measure
Baruch Y.a Lin, n.d.; Bjørnson F.O.a Dingsøyr, (2005); Boden et al. (2009); Chen C.-J.a Shih (2009); Hongmei & Huidong (2008); S. M. Jasimuddin (2007); Poort E.R.a Pramono (2009); Ren (2009); Wah C.Y. (2005); Wang & Yang (2008); Windiarti et al. (2011); Zhang Q. (2011).	Not specific, the whole process is treated but without detail.	Social capital, trustiness, communication style (virtual or face to face), commitment, sharing culture, idiom, participation, connectivity, incentives, respect, attitude, learning skills, coo-petition.
Conradi & Dybå (2001); Dan, Zhenqiang, Kaizhou, & Lei (2008); Kuk (2006); Mestad, Myrdal, Dingsoyr, & Dyba (2007); Qu G.a Ji (2011); Shou Y. (2010).	Not specific, the whole process is covered but without detail.	Communications technologies, networks size, quantity of interactions. Conversation, participation, connectivity. Written rules, procedures and degree of instruction. Communities of practice. FAQ rate of access.
Chau et al. (2003); Damian D. (2007); Hagge L. (2005); Pilat & Kaindl (2011); E Ras (2009); X Shan, Jiang, & Huang (2010).	Requirements	Perceived pay off and efficacy, common vision, group identity. Collaboration. Quantity of sharing. Cultural and physical distance. Testability.

Table built based in the literature review.

1.3 Chapter conclusions

This chapter, in order to reach this thesis objective 1, establishes what we know about KT in software engineering and factors who affect it. Also, defines a general background about KT in software engineering.

From the review done, nowadays, it was found out that the knowledge management studies have been focused on the creation and codification of knowledge, later in their socialization or dissemination ("knowledge sharing") without taking if a real KT occurs.

In the existent literature about KT in software engineering, the principal focus has been:

- KT among software development multinationals.
- KT inside development teams, using agile models.
- KT within projects in the organizations.
- KT between people within an organization.

It is found out that KT in the process of requirements elicitation has not been widely studied in the literature, just four articles were found directly related to it Hagge L. (2005); Pilat & Kaindl (2011); Salger F.a Engels (2010); Xiaohong Shan et al.(2010).

The authors mentioned above typify some factors that facilitate the KT. However they leave out any means of metric or indicator that permit to measure in any way the KT rate.

Measure the KT is a problem, because there is no clear model that allow a quantitative and/or qualitative approximation to KT, specially due to the KT is not the only way to create new knowledge (Awad, 2005; Kumar & Ganesh, 2009).

The phase of collection and specification of requirements plays an important role because is where the business needs are translated to technical language and allow setting the scope of the software project (Hagge L., 2005).

The problem of carrying the business concepts across all the software development steps has not been studied and represents an interesting percent of fail causes (23% Standish Group) in the software development projects.

Even if the software chaos report from the Standish Group has been criticized (Holtsnider et al., 2010; Jørgensen & Moløkken-Østvold, 2006), and the 189% overrun percentage in software projects has been reduced o 34% and 33%, the 68% of projects that still failing are because of the poor requirements specifications.

2. Software requirements as a process of KT.

This chapter shows a general approach to software requirements and how software requirements steps take place inside the knowledge transfer stages. Section 2.1 presents a brief description about what is known as software requirements for the purpose of this study, basically based on SWEBOOK reference. Next, section 2.2 introduces a mapping for the KT stages and SR steps, giving as a result a series of measurable factors affecting each KT-SR match. Then, in section 2.4 appears the metrics found and proposed for each group of KT-SR match. At the end, in section 2.5 conclusions of this chapter are established.

2.1 Software requirements

The requirements for a system are the descriptions of what the system should do, the services that it provides and the constraints on its operation. The process of finding out, analyzing, documenting and checking these services and constraints is called requirements engineering –RE (Kedian, Zhi, & Didar, 2008; Khan, Ahmad, & Alnuem, 2012; McGee & Greer, 2012; Wiradanti & Govindaraju, 2011). Despite heterogeneous terminology throughout the literature, RE must include four separate but related activities: elicitation, modeling, validation, and verification according to SWEBOOK. In practice, they will most likely vary in timing and intensity for different projects.
SWEBOOK states:

> Elicitation is often treated as a simple matter of interviewing users or analyzing documents, nonetheless several other elicitation methods are available. Some emphasize group sessions in the form of focus groups or workshops; others are employed primarily to elicit requirements for specific types of systems. For example, developers frequently use repertory grids, sorts, and laddering methods in specifying knowledge-based systems. Elicitation also includes those activities

that explore how software can meet organizational goals, what alternatives might exist, and how they affect various stakeholders.

Modeling: Experts have proposed many modeling methods and specification languages to make requirements precise and consistent. Traditionally these methods have separated the data, functional, and behavioral aspects of requirements and specified software by creating one or more distinct models. Prototypes, for instance, attempt to create an operational model that stakeholders can directly experience.

Validating: The purpose of validating requirements is to certify that they meet the stakeholders' intentions, try to answer if the software is being specifying the right way. Validation examines a work product (for example, a specification) to determine conformity with stakeholder needs.

Verification, on the other hand, determines whether a work product conforms to the allocated requirements, so the software is specified correctly. That is, it checks a specification for internal consistency through mathematical proofs or inspection techniques. An important point in validating and verifying requirements is prioritizing them. By addressing high-priority requirements before considering low-priority ones, you can significantly reduce project costs and duration. Moreover, throughout RE you should revisit the priorities assigned, for example, during elicitation to ensure that they continue to adequately reflect the stakeholders' needs. This highlights the recurrent nature of requirements validation and verification.

Methods for validating and verifying requirements are relatively scarce. Peer reviews, inspections, walkthroughs, and scenarios figure most prominently. Moreover, the recording of decisions and their rationales is quite useful.

Best practices in RE involve: Successful requirements engineering teams have in depth knowledge of the application domain, IT, and the requirements engineering process. In other words, successful projects have the "right combination" of knowledge, resources,

and process (Kedian et al., 2008; Khan et al., 2012; McGee & Greer, 2012; Wiradanti & Govindaraju, 2011).

Pilat & Kaindl (2011) states that stakeholder feedback plays a decisive role from the beginning to the end of successful requirements engineering projects. The most successful teams always involve customers and users in the requirements engineering process and maintain a good relationship with stakeholders.

Successful teams have an ongoing collaboration with stakeholders to make sure that requirements are interpreted properly, to deal with fluctuating requirements, and to avoid communication breakdowns. Chau et al. (2003) research supports this best practice: according to one study, user participation is one of the most important factors contributing to requirements engineering success. Successful requirements engineering teams identify the boundaries of the application domain and of the major stakeholders.

Klendauer, Berkovich, Gelvin, Leimeister, & Krcmar (2012) claims that successful projects allocate a significantly higher amount of resources to requirements engineering (28%) than the average project in this or previous field studies, and they expend these resources according to a well-defined process.

Since software requirements engineering deals with understanding of the business knowledge, KT is essential for the process success (Hagge & Lappe, 2004, 2005; Pilat & Kaindl, 2011). The result of the KT can be described as a software engineering specification, which is an official statement of what the system developers should implement. It should include a detailed specification of the system requirements.

Knowledge already acquired and codified by the requirements engineers can be given to the stakeholders for reviewing purposes. Through feedback from such reviews, more knowledge about the requirements can be transferred to the requirements engineers. This reiterates the process, and the codified knowledge put into the form of a specification improvement (Pilat & Kaindl, 2011; Xiaohong Shan et al., 2010).

During SR, two categories of knowledge are transferred and transformed. On one hand, knowledge about the requirements is being manipulated, and on the other hand,

additional knowledge about the domain for which the software is being developed is necessary (Damian et al., 2007). Stakeholders that hold knowledge about the requirements, inherently also possess domain knowledge (Xiaohong Shan et al., 2010).

2.2 KT-SR mapping

KT process stages were mapped against SR process steps. On the one hand the reference for the KT stages was taken from relevant authors according a literature review about KT in SR and their measurement, on the other hand the SWEBOOK was taken into account, for the SR process, because it is an effort from the computer society to characterize what is known about software engineering process, and promote a consistent view of SR. Finally, factors affecting KT were organized for each step mapped.

Starting with Szulanski (1996) who states that KT has four stages: Initiation, implementation, “ramp-up” and integration, other authors start using the word KT process adding or modifying steps like: information acquisition, documentation, transmission, source and receiver perception (Verkasalo & Lappalainen, 1998), gather the knowledge from a source, code it through a channel, and pass it to a receipt (Albino et al., 2004), Idea creation, sharing, evaluation, dissemination and adoption (Levine & Gilbert, 1998).

SWEBOOK divide SR in seven topics: SR fundamentals, Requirements process, elicitation, analysis, specification and validation, Practical considerations and SR tools. However, only four are going to be considered which are the related to the strictly SR process: elicitation, analysis, specification and validation.

In short, there are four dimensions for the knowledge transfer to occur according to Szulanski (1996).

1). ***Initiation***: where the decision to KT and information acquisition is done by gathering the knowledge from a source, in a software context it is supposed that the source is motivated enough to share their knowledge because the source is the client who need the software, at this point the KT for software requirements differ from classical KT in organizations, because the receiver of the knowledge (i.e. software analysts) does not intend to apply such knowledge but to build a software specification. This first step match with software elicitation stage for SR, because is where the first approach to business

knowledge stakeholders is made, those stakeholders initiate the sharing of their knowledge and KT starts.

2) ***Implementation***: is about the formal flow of knowledge from the source to the receipt, first software specification which could be seen as the source of knowledge codification occurs, the elicitation step ends and start the analysis of such first requirements, implementation cease or diminish with the software specification because is where the receipt starts using the transferred knowledge (requirements). At this point implementation step for KT differ from classical KT in organizations, because the receipt isn't going to use the knowledge in his behalf, but for a software specification analysis.

3) "***Ramp-up***": in this step initial knowledge codification and knowledge dissemination ends, software requirements are fully analyzed giving as a result the formal initiation of a software specification document. Consistence and conjecture of requirements are being evaluated. The software specification serve as a basis for agreement between customers and contractors on what the software product is to do as well and what it is not expected to do.

4) ***Integration***: begins after the receipt achieves satisfactory results with the transferred knowledge, the knowledge is adopted and the perception of source and receiver happens. In the software requirements context is about the software specification end, the awareness of needs and ambiguity are evaluated, starting and ending the validation stage of software requirements, resulting in the final software specification.

Table 3. KT in SR shows the mapping, where KT steps (there called dimensions) appears in the first column and SR stages take place in the first row, E for elicitation, A for analysis, S for specification and V for validation. In each cell appear a group of factors, named F1 to F7.

Table 3. KT in SR. Dimensions of KT vs. stages of SR.

	E	A	S	V
Initialization. *Information acquisition. *Knowledge gathering. *Knowledge sharing. *Knowledge dissemination.	F1			
Implementation. *Documentation. *Knowledge codification.	F2	F3		
“Ramp-up” *Receipt perception. *Knowledge codification. *Knowledge dissemination.		F4	F5	
Integration. *Knowledge adoption.			F6	F7

Table built based on Albino, Claudio Garavelli, & Schiuma (1998); Levine & Gilbert (1998); Szulanski (2000); Verkasalo & Lappalainen (1998) and the SWEBOOK v3.

2.3 Factors affecting each KT and SR mapping

Based on Szulanski (2000); Schwartz (2007); Minbaeva (2007); Goh, Chua, Luyt, & Lee (2008) and Simonin (2004) work, the next factors are defined for each mapping.

Factors F1, the Initialization and Elicitation are affected by the willingness to initiate transfer and propensity to share which are related to: acknowledgement and attribution, disseminative capacity, interpersonal connection and motivation of the source.

Factors F2, the Implementation and Elicitation are affected by the ease of transfer which is related to stickiness at initiation, stickiness at implementation, motivation, the awareness of need, the ability to transfer, the ambiguity of knowledge, the retentive capacity and modifiability of requirements.

Factors F3, the Implementation and Analysis are affected by the available time/access of source and receipt, reliability of the source, motivation of the receipt, ambiguity of knowledge, awareness of availability, absorptive capacity of receipt, understandability of requirements and its verifiability.

Factors F4, the Ramp-up and Analysis are affected by the requirements degree of conjecture.

Factors F5, the Ramp-up and Specification are affected by requirements internal consistency.

Factors F6, the Integration and Specification are affected by the available time/access of the receipt and source, while the awareness of need from the source and the ambiguity of knowledge.

Factors F7, the Integration and Validation are affected by the correctness and completeness of the specification.

2.4 KT-SR Metrics

This section explains an analysis over the existing metrics for software requirements seen from the knowledge transfer point of view. In order to gathering software requirements metrics a review was conducted following Kitchenham (2007) instructions using the SCOPUS database, then metrics were categorized as variables and indicators related to an aspect and dimension of analysis. Analysis dimensions were defined according to knowledge transfer stages in section 2.2, then, aspects of analysis were defined based on software requirements steps that were established in section 2.2, factors from section 2.3 and results of the literature review. Section 2.4.1 shows details of the review done and referents obtained, then, in section 2.4.2 appears the analysis of the aspects found inside each dimension of analysis and the metrics inside each aspect analysis, next, section 2.4.3 states the detailed metrics chosen for this thesis purposes, at the end section 2.5 depicts the chapter conclusions.

2.4.1 Studies selected for analysis

The literature review was performed using this terms: software requirements measurement, software requirements metrics, software requirements indicators, requirements engineering measurement; requirements engineering metrics and requirements engineering indicators".

The search equation result in 373 single papers, after reviewing their abstract, introduction and conclusions, 49 papers were selected for full reading. The criteria used for selecting those 49 papers were: the including of any mean for requirements measurement such as a variable (characteristic or attribute from an analysis unit, minimum study element, observable and measurable) or indicator (qualitative or quantitative expression observable, permits to describe characteristics, behaviors, or phenomena through some variable evolution)(Sánchez-torres, Carolina, & Torres, 2009). After the full reading for each 49 papers, 14 papers last because were the only ones who treat requirements measurement as their main issue and proposed and/or explain some sort of variable or indicator. A total of 139 metrics were found, 115 were indicators and 24 variables. Table 4 resumes the referents and quantity of variables and indicators found, and based on data from this table, Figure 10 depicts its information as percentage of variables and indicators, making clear that there are more indicators than variables.

Table 4. SR referents.

Referent	Indicator	Variable	Total
BIG EARS (Mavin & Wilkinson, 2010)	8		8
Completeness and Complexity of KAOS models(Espada, Goulão, & Araújo, 2011)	10		10
Crosscutting concerns(Conejero, Figueiredo, Garcia, Hernández, & Jurado, 2012)	2		2
Goal Oriented Quality Model(Cares & Franch, 2009)	9		9
Maintainability NFUR(Abran, Al-Sarayreh, & Cuadrado-Gallego, 2010)	3		3
Measurement Framework for Maturity RE Process(Niazi, Cox, & Verner, 2008)		14	14
Meeting Quality SR Acquisition Phase(Hanakawa & Obana, 2012)	18		18
QRE BO approach(Banerjee, Sarkar, & Debnath, 2013)	4		4
Quality of Textual Requirements(Génova, Fuentes, Llorens, Hurtado, & Moreno, 2013)	27		27

RE Initial Metrics(Costello & Liu, 1995)	7		7
Reliability NFUR(Al-Sarayreh, Abran, & Santillo, 2010)	3		3
Requirements Modifiability Management(Lam, Loomes, & Shankararaman, 1999)	4		4
SRS Metrics(Iqbal & Naeem Ahmed Khan, 2012)	16		16
Use Case Complexity(Yavari, Afsharchi, & Karami, 2011)	4	10	14
Total	**115**	**24**	**139**

Source: self-elaboration based in literature review done.

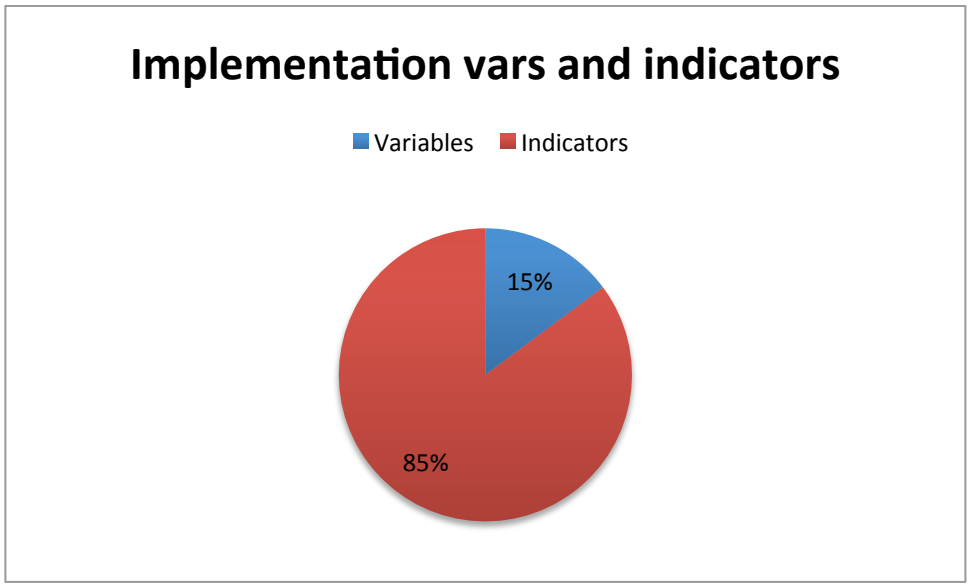


Figure 10. Implementation variables and indicators.
Source: self-elaboration based in literature review done.

2.4.2 KT-SR dimensions, aspects and metrics

Analysis dimensions are taken as KT stages defined in section 2.2. Each dimension has a series of factors affecting them as stated in section 2.3. Even when there are four analysis dimensions, in the literature review does not appear any mean of measurement for the initiation stage, here called dimension of initiation. For the other dimensions (implementation, integration and ramp-up) were found out some aspects in the review done.

So, talking about SR there is no variable nor metric found in order to take care for Factor1 - the willingness to initiate transfer and propensity to share which are related to: acknowledgement and attribution, disseminative capacity, interpersonal connection and motivation of the source- Maybe it is because in software projects contexts, it is assumed that there is a willingness to share the business knowledge, it is taken for grant that stakeholders will share their knowledge and will be able to get involved actively in the SR process, but as stated before, usually people need to get motivated in order to share their knowledge and participate (Davenport, Prusak, & Webber, 1998; Kumar & Ganesh, 2009; Szulanski, 2000).

Regarding analysis aspects, a set of elements were taken from the review and matched with factors defined above. The matching method its really simple, because it is based in the semantics of the words, for instance: it is said that ambiguity of knowledge affects the KT at the Implementation stage(Szulanski, 2000), and it is said that Ambiguity of requirements affects SR process (Mavin & Wilkinson, 2010), so we have a match. Table 5 shows the full match done.

Table 5. KT-SR Aspects.

Mapping Factor	Aspect found in KT literature	Aspect found in SR literature
F2 - Implementation and Elicitation	Stickiness at initiation Stickiness at implementation. Motivation. Awareness of need Ability to transfer Ambiguity of knowledge Retentive capacity Modifiability of requirements.	Abstraction Ambiguity Atomicity Complexity Precision
F3 – Implementation and Analysis	Available time/access of source and receipt. Reliability of the source. Motivation of the receipt. Ambiguity of knowledge,	Ambiguity Time/Access source Time/Access receipt Understandability

	Awareness of availability, Absorptive capacity of receipt. Understandability of requirements and its verifiability	
F4 – Ramp-up and Analysis	Knowledge degree of conjecture	Effort Verifiability Volatility
F5 – Ramp-up and Specification	Knowledge consistency	Verifiability
F6 – Integration and Specification	Available time/access of the receipt and source. Awareness of need from the source. Ambiguity of knowledge.	Maintainability Traceability Validability
F7 – Integration and Validation	Correctness and completeness of knowledge.	Completeness Correctness

Source: self-elaboration based in literature review done.

As a result of the factors mapping done, 16 aspects of analysis were found. Table 6, Table 7 and Table 8 shows the list of dimensions, aspects and their referents. It is interesting that most of the indicators are focused in the SR specification document per se, but as noticed by SWEBOOK, the software requirements process is more than build the SR specification; it involves an elicitation phase where the motivation and the willingness to share knowledge take place.

Another interesting result is the lack of traceability indicators. 11 authors Abran et al. (2010); Al-Sarayreh et al. (2010); Banerjee et al. (2013); Cares & Franch (2009); Conejero et al. (2012); Costello & Liu (1995); Espada et al. (2011); Génova et al. (2013); Iqbal & Naeem Ahmed Khan (2012); Lam et al. (1999); Niazi et al. (2008) state that traceability is very important, but just four of them Banerjee et al. (2013); Costello & Liu, (1995); Génova et al. (2013); Iqbal & Naeem Ahmed Khan (2012); Niazi et al. (2008) define any mean to measure the traceability. Traceability is important because traceability enables to perform an effective control of the requirements, and due to requirements

changes constantly in every real word software project, traceability is important to keep control over software scope.

In the next page (landscape), Table 6 will show the number of variables and indicators found for implementation dimension. As there could be seen in Figure 11, most of indicators are related with *understandability*, *ambiguity* and *complexity,* that tendency in the referents are related with SR process difficulties found by authors (Espada et al., 2011; Génova et al., 2013; Hanakawa & Obana, 2012). On the one hand, we could say that the most important aspect related with KT implementation dimension is the understandability, this makes sense because in this dimension, requirements elicitation is finalized and requirements analysis begin, so requirements should be fully understood. On the other hand, *time* and *abstraction* are not widely explored, but they are important indicator for the requirements process.

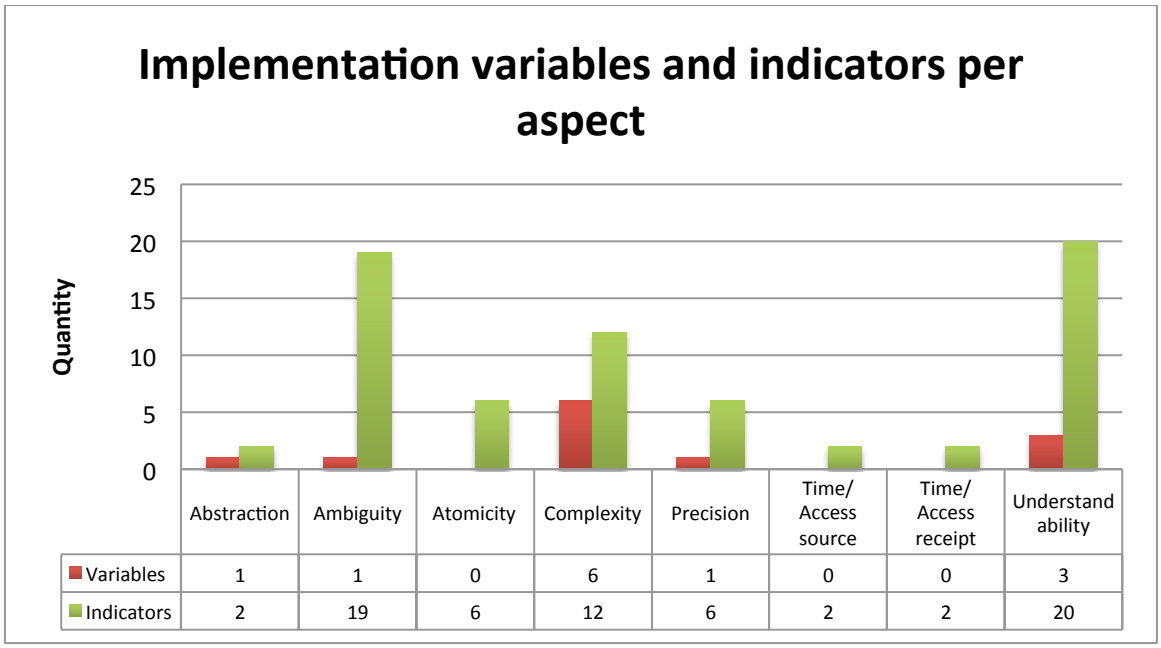


Figure 11. Implementation variables and indicators per aspect
Source: self-elaboration based in literature review done.

Table 6. Implementation dimension, their referents and aspects.

Referent	Abstraction		Ambiguity		Atomicity		Complexity		Precision		Time/Access source		Time/Access receipt		Understandability	
	V	I	V	I	V	I	V	I	V	I	V	I	V	I	V	I
Measurement Framework for Maturity RE Process	1		1						1						3	
Quality of Textual Requirements		2		3		6				5						7
BIG EARS				2			1			1						1
Goal Oriented Quality Model				3								1		1		2
Meeting Quality SR Acquisition Phase				9												9
QRE BO approach				1												
SRS Metrics				1												1
Completeness and Complexity of KAOS models								8								
Maintainability NFUR								1				1				
Reliability NFUR								1						1		
Use Case Complexity							5	2								
Total	1	2	1	19	0	6	6	12	1	6	0	2	0	2	3	20

Note: V:Variable I:Indicator

Source: self-elaboration based in literature review done.

Next, from Table 7 we can conclude that traceability and completeness take the most important roles within integration dimension. Since integration dimension deals with analysis and specification, traceability is very important as a key to do validation over the requirements towards other development phases like design, code and testing, likewise backwards from those phases to requirements.

Validability takes the last place based on their quantity of indicators; validation should be taken seriously, because in integration dimension the validation of requirements is made.

Table 7. Ramp-up dimension, their referents and aspects.

Referent	Completeness		Correctness		Maintainability		Traceability		Validability	
	V	I	V	I	V	I	V	I	V	I
BIG EARS		1		1						
Completeness and Complexity of KAOS models		2								
Goal Oriented Quality Model		2								
Measurement Framework for Maturity RE Process	1		1		2		2		1	
QRE BO approach		1		1				1		
RE Initial Metrics		2						2		
SRS Metrics				4		2		4		3
Crosscutting concerns						2				
Requirements Modifiability Management						3				
Quality of Textual Requirements								2		1
Total	1	8	1	6	2	7	2	9	1	4

Note: V:Variable I:Indicator

Source: self-elaboration based in literature review done.

In the next page, Table 8 shows the last list of indicators found. The ram-up dimension, different from traditional KT, in SR could not be said that is about to apply knowledge learnt, but, to specify a document based on a plenty of knowledge about requirements, because ramp-up is mapped with analysis and specification, volatility must be controlled and verifiability is a must in order to do a good analysis. Also could be noticed, that effort

aspect have more variables than indicators, it could be related with the lack or indicators for software estimating.

Table 8. Integration dimension, their referents and aspects.

Referent	Effort		Verifiability		Volatility	
	V	I	V	I	V	I
Maintainability NFUR		1				
Reliability NFUR		1				
Use Case Complexity	5	2				
BIG EARS				1		
Measurement Framework for Maturity RE Process			1			
Quality of Textual Requirements				1		
SRS Metrics				1		2
RE Initial Metrics						1
Requirements Modifiability Management						1
Total	5	4	1	3	0	4

Note: V:Variable; I:Indicator

Source: self-elaboration based in literature review done.

2.4.3 Discussion

KT for SR could not be seen as a typical KT process, because the receipt of the knowledge does not intend to apply the knowledge gathered from the source rather than understand and code it in a SR specification. For instance: in the context of a SR process for a software about judicial collection, a receipt (software engineer or requirements analyst) does not intend to apply or mimic the knowledge gathered from a source (lawyer or accountant) about a judicial collection process, the receipt must to understand the receipt knowledge and then translate it to a SR specification or two, one for developers and other for stakeholders (maybe the same source of knowledge) validation.

Another difference, in the context of KT for SR, is in the source of knowledge intentionality, because for a SR process, the source does not pretend to transfer his knowledge used in his day to day work, but, the source intends to transfer needs about a software that have to fit some business rules or logic the source is expert in.

2.5 Chapter Conclusions

This chapter was developed in the interest of this thesis objective 2, because characterize the KT process in the requirements elicitation process. Furthermore, this chapter maps the relations within KT and SR processes activities.

Knowledge transfer for software requirements is different from classical approach to KT, because transfer of a full body of knowledge is not intended, instead of that, need who relate with some business knowledge are transferred.

A mapping from KT and SR were made based on their sub process similarities, in order to find out any mean of indicators. 139 variables and indicators were found out in the literature.

One mapping last without indicators, the one regarding factor 1 Initialization and Elicitation. So, some measurable aspects are needed for it.

3. Proposal Methodology for KT measurement in SR

This chapter shows the stages series to build the measurement methodology. The methodology will consider: dimensions and aspects of analysis, the identification of indicators and variables from chapter 2, and, will include a stage to find out a set of indicators who apply to a certain case, this is done because of organizations diversity, due to such diversity, the full set of indicators from chapter 2 could not fit all organizations reality, so a previous validation must be made. This chapter is organized in two parts, first part (section 3.1) outlines the purpose, scope, end users and replicability of the proposal methodology, and meanwhile, second part (section 3.2) details the proposal methodology as well, whose sections include: the initial framework, the suggestion to make indicators refinement, how to conduct the data collection and analysis, next, analysis o findings appear and at the end the feedback appears.

3.1 Overview of the proposal methodology

The proposal methodology serves as a recommended mean to achieve the KT measurement in SR. According Avison & Fitzgerald (2006), the recommended means usually includes the identification of phases, procedures, tasks, rules, techniques, guidelines, documentation and tools. They might also include recommendations concerning the management and organization of the approach and the identification and training of the participants.

Since there is no much knowledge about KT in SR, the proposal methodology is kind exploratory, so based on the review done, indicators are going to be gathered and will be refined inside an organization, furthermore data collected by questionnaires will be analyzed for validity and reliability.

In next section, we will precede with proposal methodology general aspects, including the purpose, scope, end user and the methodology replicability.

3.1.1 Purpose of study

The purpose is to answer the research question How to measure KT in SR? To do this, the previous KT-SR characterization will be used in order to validate which stages fit or not in a real environment.

3.1.2 Scope of the proposal methodology

This is an exploratory methodology because of the lack of literature regarding KT measurement. Likewise, we want to explore the current status of SR inside an enterprise with real software projects, not only in terms of the literature indicators and stages, but also inquire about the organization perceptions as well as their views on how the KT affects the SR process.

3.1.3 End users

Software development organizations and any SR or KT enthusiasts.

3.1.4 Replicability

Taking into account that organizations and people use to change and evolve, replicability could be affected, so rather than replicate the results, this proposal methodology could be replicated though the application of methods for data collection and analysis.

3.2 Proposal methodology

Departing from (Avison & Fitzgerald, 2006) proposal methodology definition, this section presents the procedures, techniques, tools and help documentation designed in order to realize the KT measurement in SR.

3.2.1 Scope

This proposal methodology serves as guide in order to find out a way to measure the knowledge transfer in the software requirements process inside an organization, so next sections state the steps to pursue such goal.

3.2.2 Initial framework

In the past chapter, a set of 139 indicators and variables were found out. Since it would be nice to apply the full set of indicators, it would be too heavy to survey. So, from each one of 16 analysis aspects, two indicators will be selected.

Next the 16 aspects appear. For abstraction aspect Table 9, there is only one referent so it is used.

Table 9. Abstraction indicators.

	Abstraction	The requirements tell what the application must do without telling how it must do it, i.e., excess of technical detail about the implementation must be avoided in the specification of the requirements.
#	Referent	Indicator
1	Quality of Textual Requirements	# Control flow terms: while, if then, when. Count one per requirement/user story.
2	Quality of Textual Requirements	# Design Terms: (technology or design related) method, parameter, database. Count one per requirement/user story.
3	Quality of Textual Requirements	At the end abstraction degree = # req abstract / # req

Source: (Génova et al., 2013)

For the ambiguity aspect Table 10, there are 6 referents, but only Génova et al. (2013) and Iqbal & Naeem Ahmed Khan (2012) are not tool or method dependent.

Table 10. Ambiguity indicators.

	Ambiguity	There exists only one interpretation for each requirement (unambiguity and understandability are interrelated, they could be even the same property; some authors have coined the term ‘‘uniguity”
#	Referent	Indicator
4	Quality of Textual Requirements	# Imprecise terms
5	Quality of Textual Requirements	# Overlaping req
6	SRS Metrics	Ri/Rt x 100, < 95% ambiguous Ri= requirements having the same interpretation Rt = total requirements

Source: (Génova et al., 2013) and (Iqbal & Naeem Ahmed Khan, 2012)

For the atomicity aspect Table 11, there is only one referent.

Table 11. Atomicity indicators.

	Atomicity	Each requirement is clearly determined and identified, without mixing it with other requirements.
#	Referent	Indicator
7	Quality of Textual Requirements	# Req size <= 18 words.
8	Quality of Textual Requirements	# Req dependencies (other reqs, artifacts), max 3 dependencies

Source: (Génova et al., 2013)

For the complexity aspect Table 12 (Yavari et al., 2011) is selected because others are tool dependent or method dependent.

Table 12. Complexity indicators.

	Complexity	Degree of interrelationships of requirements and actors or goals/objects.
#	Referent	Indicator
9	Use case complexity	Level of actor complexity: Simple if An API or program interface, score 1. Average if Network protocols such as TCP/IP, HTTP, score 2. Complex Graphical User Interface, score 3.
10	Use case complexity	Level of UseCase complexity: Simple if 1 to 3 transactions, score 5 . Average if 4 to 7 transactions, score 10. Complex More than 7 transactions, score 15.

Source: (Yavari et al., 2011)

For precision aspect Table 13, there are only two referents who has indicators.

Table 13. Precision indicators.

	Precision	All used terms are concrete and well defined.
#	Referent	Indicator
11	Quality of Textual Requirements	# anaphorical terms. They are typically personal pronouns (it), relative pronouns (that, which, where), demonstrative pronouns (this, those), etc. Max 2 per req/story.
12	Big Ears	Vagueness, #req that are vague / # total req. A requirement is vague if require more dialog to understand?

Source: (Génova et al., 2013; Mavin & Wilkinson, 2010)

Time indicators are mentioned here, Table 14 and Table 15, but discarded because (Hanakawa & Obana, 2012) makes and extensive explanation about time as a transversal enabler for SR quality. So, for the purpose of this thesis, the more time involved in SR the better SR quality.

Table 14. Time/Access source indicators.

	Time/Access source	Quantity of time allocated for the knowledge receipt to interact with the knowledge source.
#	Referent	Indicator
13	Goal oriented quality model	Average over the total amount of time Velocity of agreement that the group has arrived to an As-Is on domain models model over the 80% of agreement in past modeling activities.
14	Maintainability NFUR	Schedule in Months = 3.0 * person-month^1/3

Source: (Abran et al., 2010; Cares & Franch, 2009)

Table 15. Time/Access receipt indicators.

	Time/Access receipt	Quantity of time allocated for the knowledge source to interact with the knowledge receipt.
#	Referent	Indicator
15	Maintainability NFUR	Mean time to modify a requirement.
13	Goal oriented quality model	Average over the total amount of time Velocity of agreement that the group has arrived to an As-Is on domain models model over the 80% of agreement in past modeling activities.

Source: (Al-Sarayreh et al., 2010; Cares & Franch, 2009)

For the understandability aspect Table 16, there are 6 referents, but only (Génova et al., 2013) and (Iqbal & Naeem Ahmed Khan, 2012) are not tool or method dependent.

Table 16. Understandability indicators.

	Understandability	The requirements are correctly understood without difficult
#	Referent	Indicator
16	Quality of Textual Requirements	Readability index: RFlesch = 206.835 - (1.015 9 WPS) - (84.6 9 SPW), where WPS and SPW stand respectively for average words per sentence and average syllables per word. RKinkaid = 0.39 9 WPS ? 11.8 9 SPW - 15.59. This index yields a result between 0 and 12.
17	SRS Metrics	Number of req that are understandable to all the users and reviewers. Ru/Rt x 100 Ru = Req understood by users Rt = Tot req

Source: (Génova et al., 2013) and (Iqbal & Naeem Ahmed Khan, 2012)

For completeness aspect Table 17, (Costello & Liu, 1995; Espada et al., 2011) referents are selected, because others are tool or method dependent.

Table 17. Completeness indicators.

	Completeness	It is a quality factor, which means necessary requirements objects are not lacking specification.
#	Referent	Indicator
18	RE Initial metrics	Are all allocated higher-level requirements are addressed?
19	Completeness and Complexity of KAOS models	Percentage of leaf goals/req that have an associated agent/actor.

Source: (Costello & Liu, 1995; Espada et al., 2011)

For correctness aspect Table 18, (Iqbal & Naeem Ahmed Khan, 2012; Mavin & Wilkinson, 2010) referents are selected, because others are business object dependent.

Table 18. Correctness indicators.

	Correctness	It is a quality factor which means how many requirements in requirement specification meet customer's need
#	Referent	Indicator
20	SRS Metrics	Rc /Rt x 100 Rc = req having the same interpretation Rt = Tot req shoudl be >= 80%
21	Big EARS	Are all process consistent (knowledge holder-receipt)

Source: (Iqbal & Naeem Ahmed Khan, 2012; Mavin & Wilkinson, 2010)

For maintainability aspect Table 19, there are 3 referent, but (Iqbal & Naeem Ahmed Khan, 2012; Lam et al., 1999) referents are selected, because even when crosscutting concerns (Conejero, Figueiredo, Garcia, Hernández, & Jurado, 2009) are an interesting concept, there are no clear example about how to use it.

Table 19. Maintainability indicators.

	Maintainability	It is a quality factor witch means how requirements change and stabilize over the time.
#	Referent	Indicator
22	Requirements modifiability management	% Acceptance rate. The acceptance rate is the % of requirement changes accepted by the customer at delivery time within a given reporting period.
23	SRS Metrics	% of the requirements changed during each phase of system development. Rch = total requirements to be changed Rc = total correct requirements

Source: (Iqbal & Naeem Ahmed Khan, 2012; Lam et al., 1999)

For traceability aspect Table 20, there are 4 referents, but all of these just make modifications reference to a initial source which is (Costello & Liu, 1995) RE Initial metrics.

Table 20. Traceability indicators.

	Traceability	Indicates degree to which development organization maintains accountability for meeting requirements at each stage of life cycle via a requirements traceability matrix (RTM). Provides quantitative means for determining whether all required relationships/dependencies are addressed.
#	Referent	Indicator
24	RE Initial metrics	# of req traced between spec, design and test / # of tot req
25	RE Initial metrics	# of req in the RTM / # of req

Source: (Costello & Liu, 1995)

For validability Table 21, there are only two referents.

Table 21. Validability indicators.

	Validability	The client must be able to confirm (validate) that the requirements effectively express the system that answers his or her needs.
#	Referent	Indicator
26	Quality of textual requirements	# of req versions (max 2)
27	SRS Metrics	# of req validated / # of req

Source: (Génova et al., 2013; Iqbal & Naeem Ahmed Khan, 2012)

Effort aspect, Table 22, will not be considered because its more related with estimations done over requirements than the SR process as well.

Table 22. Effort indicators.

	Effort *	
#	Referent	Indicator
28	Maintainability NFUR	Effort in Person Month = FP divided by no. of FP's per month.
29	Use Case complexity	Number of Main and Alternative Scenario

Source: (Al-Sarayreh et al., 2010; Yavari et al., 2011)

For the verifiability aspect Table 23, there are 4 referents, but only (Génova et al., 2013) and (Iqbal & Naeem Ahmed Khan, 2012) are not tool or method dependent.

Table 23. Verifiability indicators.

	Verifiability	The engineer must be able to check (verify) that the produced system meets the specified software.
#	Referent	Indicator
30	Quality of textual requirements	# req versions (max2) + # req dependencies (max 9)
31	SRS Metrics	# req verified ok / # req verified

Source: (Génova et al., 2013) and (Iqbal & Naeem Ahmed Khan, 2012)

For volatility aspect Table 24, there are 3 referents, but (Costello & Liu, 1995; Lam et al., 1999) is chosen because the other (Iqbal & Naeem Ahmed Khan, 2012) are based on the first two.

Table 24. Volatility indicators.

	Volatitlity	Indicates changes (additions, deletions, modifications) and reasons for changes to requirements. Provides insight into system maturity and stability.
#	Referent	Indicator
32	Requirements modifiability management	# req modified, added and removed / # total req
33	RE Initial metrics	# req defects, faults / # req

Source: (Costello & Liu, 1995; Lam et al., 1999)

Additionally we are going to define some others aspects to cover the first dimension of KT (initiation), because there were no metrics found regarding with it.

The new aspects are related to factor F1 where the Initialization and Elicitation are affected by the willingness to initiate transfer and propensity to share which are related to: acknowledgement and attribution, disseminative capacity, interpersonal connection and motivation of the source as shown in Table 25, likewise in Table 26 appears the proposed questions to make the motivation measurement.

Table 25. Aspects proposed, Factor 1 Initialization and Elicitation.

Factor	KT aspect	SR proposed aspect
F1	Acknowledgement and attribution. Disseminative capacity. Interpersonal connection. Motivation of the source.	Propensity to share from the knowledge holder. Disseminative capacity from the knowledge holder. Requirements analyst Interpersonal connection. Acknowledgement and attribution of the knowledge holder. Knowledge holder motivation.

Proposal built based on (Schwartz, 2007; Szulanski, 1996, 2000) and the SWEBOOK v3.

Table 26. Motivation indicators.

	Motivation	Incentive that encourage knowledge transfer.
#	Index	Questions
34	Sharing motivation index	I'm happy with my experience so I Want others to experience it? I want to share my great experience? I want to help with my positive experience? I want to save others from having same negative experience as me? I am unhappy with the current process the software is automating?
35	Propensity to share index	I feel good when tell others about my success cases? I want to get rewarded for sharing my expertise? I like talking to people with similar interest?

Questions based on (Roy, 2011)

3.2.3 Indicators refinement

Even when we already have an initial set of indicators, we still need to refine indicators in spite of identify essential statements about organizations reality as follows: 1) which software requirements phases are used in the organization? 2) Which KT aspects match the SR process in the organization? And, 3) do the indicators fit the software requirements process inside the organization? To do the refinement a kick off meeting should be applied to a team involved in the software requirements process, within the team people should be: software managers, team leaders, software requirements analysts, developers and business knowledge holders if possible.

The kickoff meeting is suggested, because in such meetings team roles are discussed and the model could be presented, so the three questions above can be debated too.

3.2.4 Data collection

On the one hand, the proposed method for data collection is a questionnaire, since is the method used for most of the studies reviewed as state in Table 1. On the other hand, even when this is an exploratory research to understand how KT happens in SR and this way find how to measure it; a questionnaire is needed in order to get a quantitative insight of KT measurement.

Type of questionnaire: The questionnaire is type transversal (data will be collected in a specific time).
Questionnaire elements: The questionnaire designed should have five components:

- The *first* a set of questions to characterize the respondents,
- A *second* about SR quality attributes,
- The *third* about SR relation with performance (process delay and software errors),
- A *fourth* to re test SR quality attributes,
- And the *fifth* is dedicated to knowledge transfer related questions.

Questionnaire items scale: The scale used in the first component is nominal (categorical or discrete), in order to categorize respondents. The others components are Likert type, because that way indicators defined in section 3.2.2 Initial Framework, and to favor correlation analysis.

Target audience: People involved in software development, specially with SR. There is no sampling, because generally, a software development teams are small, so all team members will be considered.

The instrument: The instrument is built using answers taken from previous step, apply a survey in order to gather indicators regarding people (e.g. people motivation) and apply indicators over artifacts owners (e.g. SRS completeness). The survey should include questions defined for each referent; as well as quantitative indicators should use referent's formulas defined in section 3.2.2 Initial Framework. The instrument analysis and validity will be discussed in next sections.

Survey was build based on indicators above in section 3.2.2. Questions was grouped in five groups as follows: 1) People who respond characteristics, in order to get the perception among different roles in software development; 2) Basic Requirements quality attributes: understandability related attributes (complexity, ambiguity, atomicity, precision), completeness, traceability, correctness in order to get and insight in their influence over the SRS compliance; 3) SR influence in project time delay and errors; 4) KT related questions about motivation and willingness to knowledge sharing/transfer; and 5) Re test Requirements quality attributes, focused in how changeability (due to understandability,

abstraction) affects user validability (user acceptance). At the end, appears a question about How could knowledge transfer in software requirements be improved? Just to figure out respondent's perception about KT in SR. Figure 12 depicts the survey structure.

The purpose is to search for correlation between each of the five groups, especially against KT.

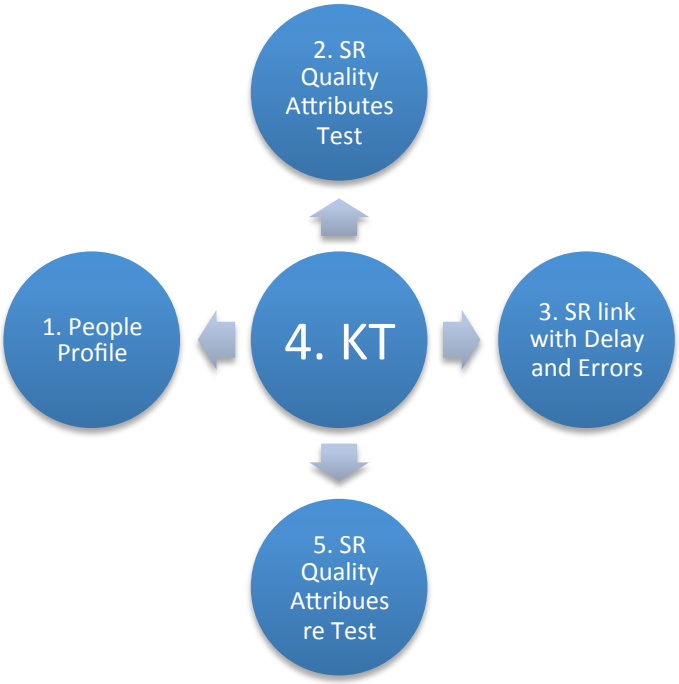


Figure 12. Survey Structure. Source: Self-elaboration.

Table 27 displays the questions and possible answers for each survey's groups. For each group of questions, Likert and numeric range answers were designed according section 3.2.4 an developed follows: for group 1 (people profile): numeric ranges; for group 2 (SR Quality Attributes Test) and 5 (SR Quality attributes re Test), four level frequency options; for group 3 (SR link with Delay and Errors) and 4 (Knowledge transfer), four level agreement options.

Table 27. List of questions for groups 1 to 5.

Group 1 (People Profile)	**Group 2 (SR Quality Attributes Test)**	**Group 3 (SR link with delay and Errors)**	**Group 4 (Knowledge transfer)**	**Group 5 (SR Quality attributes re Test)**
Possible answers: numeric ranges.	*Possible answers: Four level frequency options (Always, Often, Some Times, Never)*	*Possible answers: Four level agreement options (Totally Agree, Agree, Partially Disagree, Totally Disagree)*	*Possible answers: four level agreement options (Totally Agree, Agree, Partially Disagree, Totally Disagree)*	*Possible answers: Four level frequency options (Always, Often, Some Times, Never)*
What is your principal rol at your organization / team work?	There is just one single interpretation for each requirement.	Requirements cause delay or failures in software projects.	I feel happy with my experiences, so I want others to know them.	Are correctly understood, without difficulty.
How many people work in software requirements at your company?	Each requirement is clearly determined and identified, overlapping among requirements.	Requirements may cause a poor software quality (stability, scalability, completeness, etc.)	I want to share my huge experience.	Are specified without explain technical details about implementation.
How many years experience have the people working with requirements elicitation at your organization?	Terms used in requirements are concrete and well defined.		I want to help others with my positives experiences.	Meets users needs.
How many actors, extension points or dependencies, does a requirement have?	All requirements and their dependencies are specified.		I want to save others from bad/negative experiences I've had.	Stop changing over time.
	There are compliance monitoring for requirements, at every		I don't feel happy with	Change due to failures

	Users state that requirements express their needs effectively.		I feel good when talking with others about my successful experiences.	Change because of user validations.
	Software produced fulfills requirements specification.		I want to be rewarded for sharing my knowledge.	
			I like to talk with people with common interests.	

Source: Self-elaboration. Based on initial framework defined in 3.2.2.

3.2.5 Data analysis

Reliability and validity tests over the data collection methods should be used.

Validity is concerned with the accuracy of our measurement, and it is often discussed in the context of sample representativeness. However, validity is also affected by survey design since it also depends on asking questions that measure what we are supposed to be measuring (Mora, 2011; Walonick, 2012).

Reliability, on the other hand, is concerned with the consistency of our measurement, that's the degree to which the questions used in a survey elicit the same type of information each time they are used under the same conditions.

Reliability is also related to internal consistency, which refers to the degree different questions or statements measure the same characteristic. This can be tested by using correlations, split sample comparisons or methods such as Cronbach's Alpha (Mora, 2011; Walonick, 2012).

According Walonick (2012) reliability is synonymous with repeatability. A measurement that yields consistent results over time is said to be reliable. When a measurement is prone to random error, it lacks reliability. The reliability of an instrument places an upper limit on its validity. A measurement that lacks reliability will also lack validity. Three methods to test reliability are suggested: test-retest, equivalent form, and internal consistency.

3.2.6 Analysis of findings

Once survey analysis is done, its time to make a correlational analysis using linear regression, it is supposed that the more the KT the best the SR. Additionally, make a significance analysis using t-student distribution.

At the end ask team people involved in SR process, about the perception about the KT SR methodology used.

3.2.7 Proposal Methodology Feedback

The proposed methodology strengths and weaknesses should be detected by applying the methodology to a case study, as well as the preliminary results for the indicators and the general profile.

Results obtained for each proposed indicator, as well as the strengths and weaknesses detected when applying the model, must became feedback to adjust the model. It's supposed that it would be iteratively applied until all the indicators and profiles were refined (Sánchez-torres et al., 2009).

3.3 Chapter conclusion

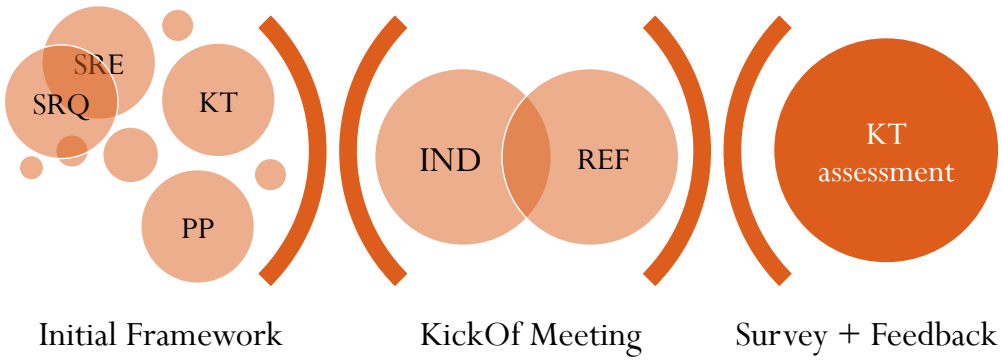
This chapter was developed to achieve this thesis objective 3, design a proposal methodology for KT measurement in SR.

Since there are a lot of indicators taken from the literature, only two indicators was selected per analysis dimension in order to build the initial set of indicators. Those indicators establish the initial framework that is going to be refined when conducting the study case.

Some indicators were discarded because were more related with software requirements specifications rather than software requirements process.

Survey's data analysis was defined in order to grant certain measure of internal consistency, and then correlational analysis was selected as the statistical tool for indicators evaluation.

Figure 13 depicts the proposal methodology, where tree stages appears, the first one comprises the instrument building, there appears the four elements defined (SRQ software requirements quality attributes, SRE software requirements re test, KT component and PP for people profiles), the second one is about the instrument indicators refinement through a kickoff meeting, and the last one is the KT assessment through the survey and the statistical tools.



Initial Framework	KickOf Meeting	Survey + Feedback
16 Indicators 4 dimensions	1. which software requirements phases are used in the organization?; 2. Which KT aspects match the SR process in the organization?; and 3. do the indicators fit the software requirements process inside the organization?	Likert Mean Index IC Correlation

Figure 13. The proposal methodology.
Source: self-elaboration based on proposal methodology.

4. Study Case

This chapter presents how the study case was conducted and shows the obtained results in addition to the proposal methodology feedback. Section 4.1 introduces the case, afterward methodology was applied as defined in chapter 3; thereafter, the indicators refinement is done in section 4.2, which was made using the initial framework defined in section 3.2.2; then, the data collection was made in section 4.3 according to what was defined in section 3.2.4; after that, data analysis was performed in section 4.4 following the instructions given by section 3.2.5; next, the analysis of findings was achieved in section 4.5.2.1 consistent with guidelines detailed in section 3.2.6; afterwards, the proposal methodology feedback is presented in section 4.6 as said by section 3.2.7; and finally section 4.7 shows this chapter conclusions.

Data collection and their respective analysis presented in this chapter serves as an executive summary, data analysis was done using aggregated data, so, this chapter doesn't presents details about answers gathered and the full calculations used in the analysis, the questionnaire details appears in Appendix A and B.

4.1 Case introduction

The case was developed at organization ABC (real name is reserved for privacy policies), this organization is a holding which groups more than ten enterprises and have almost fifty four million customers in Brazil, ABC organization is also present in nearly seventy countries. ABC organization is dedicated to the humanitarian help, was founded in 1996 and currently has more than 1000 employees just in São Paulo.

The case will be focused at the fundraising process, which is done by the technology, logistics and operations team who serves three ABC's enterprises. The technology team

is responsible for providing the software to run a fundraising of almost 40 million dollars per year.

There are fifteen 15 people working at the tech team, but for this case are nine (9) people the selected to participate, who are engineers and analysts, five (5) of them have more than 10 years experience in software development.

4.2 Indicators refinement through Kickoff meeting

According to the methodology defined in chapter 3, an initial kickoff meeting was done, in order to present the initial measurement indicators and discuss and answer the following questions: 1. which software requirements phases are used in the organization?; 2. Which KT aspects match the SR process in the organization?; and 3. do the indicators fit the software requirements process inside the organization?

4.2.1 People involved in the kickoff meeting

They were 5 senior software engineers with more than 10 years experience, 3 of them are responsible for core businesses requirements gathering, analysis and specification. The remaining two are dedicated to software architecture and requirements validation. They were also 4 full time dedicated analysts and developers.

4.2.2 The organization's software requirements process

The question: which software requirements phases are used in the organization? was discussed, and the whole agree that there exists an initial meeting with the knowledge owner, were general goals are depicted, then functionalities are decanted and finally a document is done. After the document is done, a verification phase exists to grant technical viability, then, the specification is published in a collaboration site and assigned to a developer.

So, it's seen that classical 3 initial phases approach from SWEBOK v3 is followed, but client/ business knowledge owner /stakeholder validation is not done.

Even when validation was done in somewhere between elicitation and specification, they all agree that validation should be included as a final phase, and were noticed that lack of validation already costs a five years failed project, after the five years the software did not satisfy users/business real needs.

4.2.3 The KT aspects in the organization SR.

The four knowledge transfer stages defined in chapter 2 were presented. And they agree that the 3 initial stages are done, but disagree about stage 4 integration, because they argue that there is no need for knowledge receipt to apply the knowledge gathered, rather than be able to explain and/or code it. Nevertheless was explained that KT integration, in software requirements doesn't mean apply the knowledge learnt, rather than specify and verify it.

4.2.4 The framework defined

Since there does not exist any measurement for requirements, the team agree in apply the full initial set of indicators, even when the validation stage does not exists as well in the current SR process, but, it is embedded in the specification stage.

4.3 Data Collection

An online survey was designed and published using Google forms. The survey has 5 groups of questions as Figure 12 states. The instrument was applied as defined in section 3.2.4. The survey was applied to the same group of people interviewed at the kickoff meeting. Responses gathered appears in appendix A.

4.4 Data analysis

First of all, for each measure a numeric weight was defined, then Cronbach's alphas were calculated, next an index for each group was defined and based on such indexes correlation analysis was performed.

4.4.1 Measures and Weights normalization

The normalization done was linear; each item in the survey receives the same weight. Table 28 shows the measures normalization done for this survey.

Table 28. Survey Measures and Weights.

Numeric Range Measure	Weight defined
Less than 3.	1
From 3 to 5.	2
From 6 to 10.	3
More than 10.	4
Frequency Measure	Weight
Never	1
Some times	2
Often	3
Always	4
Agreement Meausure	Weight
Totally disagree	1
Partially disagree	2
Agree	3
Totally agree	4

Source: self-elaborated.

4.4.2 Internal consistency

On the one hand, alphas gathered show that there is an acceptable internal consistency for every group except the group 3. Even when the alpha is almost 0.8, the second questions about software quality have a big deviation, so there is no clear consensus about the implications of requirements per se in software quality, it seem that people surveyed agree that is more about the process than requirements who affect software quality. On the other hand there is a great consistency for KT related questions in the survey, that's very interesting and one could make an initial forecast saying that there is a strong KT influence in SR.

In table Table 29 appears the Cronbach's alpha for internal validation purposes, it's noticed that for group 1 there is no alpha calculations due to we expect high variability in those answers. Annex 0 shows the full set of answers gathered.

Table 29. Cronbach's Alpha for groups 2 to 5.

Group	Cronbach's Alpha
2. Requirements quality.	0.842975207
3. SR and time delay/errors.	0.786885246
4. KT elements	0.93407639
5. Requirements quality re test.	0.833333333

Source: Self-elaborated.

4.5 Resume of findings.

This section displays aggregated information in order to explain major implications found. To do that, all responses have been summarized in their correspondent groups of questions, according what was presented in section 3.2.4. Based on data from those groups of questions, correlational analysis is performed. This results are organized as follows, first the principal disaggregated findings appears in section 4.5.1, there, each indicator findings is depicted, then, in section 4.5.2, aggregated findings are explained, here, group of indicators findings are presented.

4.5.1 Indicator's findings

From data gathered, next, appear fundamentals findings organized by group of questions. The first one is SR Quality (questions 5 to 11), in Figure 14, appears interesting correlations. On the one hand Its seen that *complexity, ambiguity and atomicity* reach similar values, and that impacts *completeness, and traceability*, what makes sense, since poor handled complexity, ambiguity and atomicity, leads to hardness completeness and traceability for SR. On the other hand its see that *precision* indicator have similar values with *correctness*, what could be related with the SRS compliance, but, due to lower values in other indicators, its seen that SRS is accomplished but with overruns.

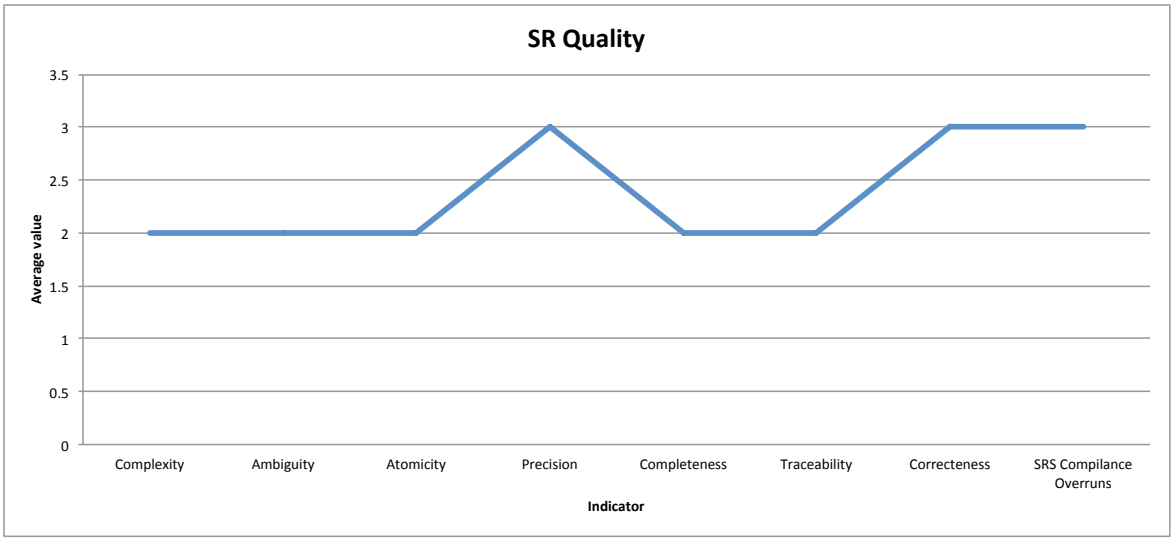


Figure 14. SR Quality Indicators results.
Source: Self elaborated based on data gathered.

Continuing with second (questions 12-13, indicators about SR and time/delay errors aka KT ERR and indicators about SR poor specification aka KT QOS) and third group of indicators (questions 14 -21, indicators about KT) Figure 15 depicts results found out about requirements influence in project errors and delays, along with KT. There appears that requirements per se does not influence project errors or delays, overruns are more correlated with the first group of indicators at Figure 14. But, an interesting finding, as can be seen in figure below, is that *the more KT the less SR errors and delays*.

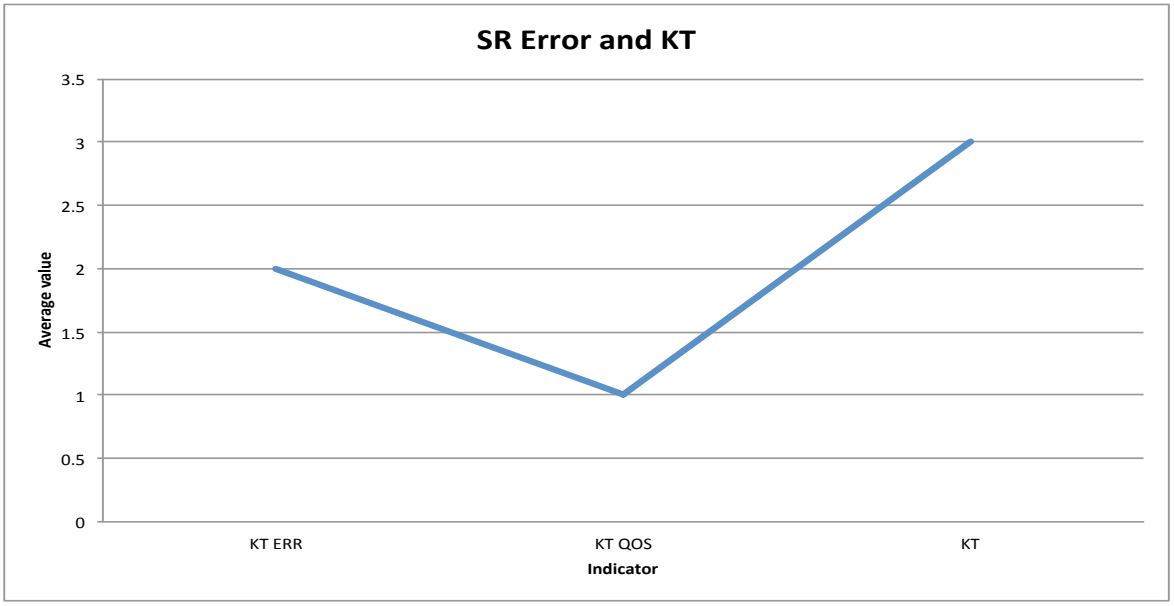


Figure 15. SR Error and KT indicators values.
Source: Self elaborated based on data gathered.

Last group of indicators (questions 22 – 27, indicators about understandability, abstraction, validability, maintainability as "changeability over time" and volatility, due to errors and due to user validations) values are depicted by Figure 16, there can be appreciated that no matters the good *abstraction and validability* requirements does not stop changing over time, and if requirements change over time is due to user validations.

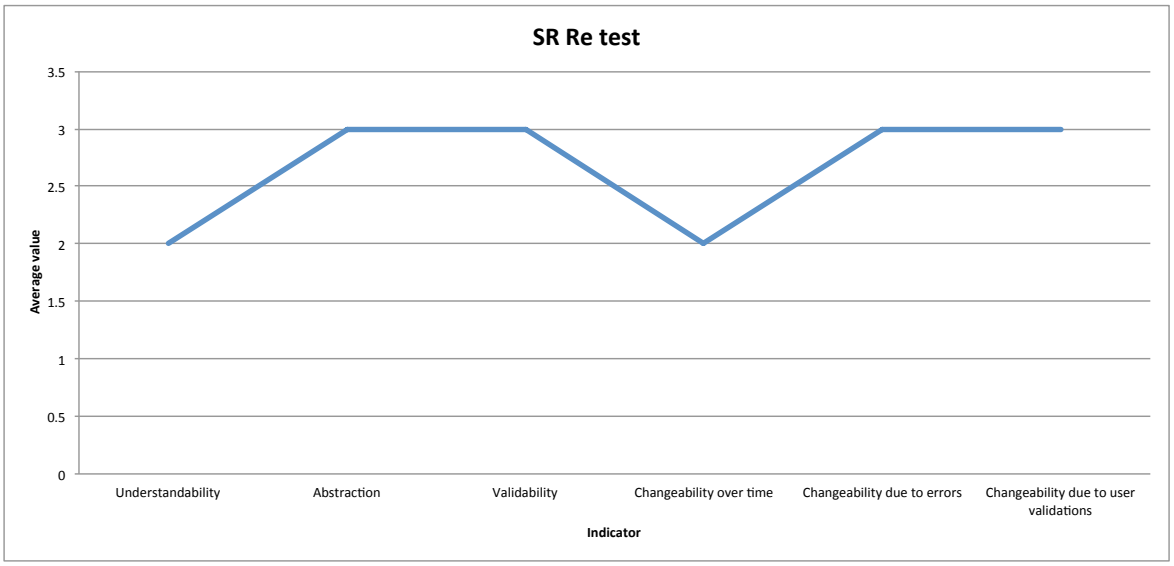


Figure 16. SR Re test indicators values.
Source: Self elaborated based on data gathered.

4.5.2 Group of indicator's findings

Once all the answers were normalized according 4.4.1, it was possible to calculate the average value (arithmetic mean) for all respondents' answers, in each questions group. The resulting value was rounded, so there are no decimal points. These results are displayed in Table 30. The mean values for each group will be used to make correlational analysis between the groups. Each value in Table 30 represents the arithmetic mean value, for one respondent, in each group of questions.

Table 30. Respondent answer's mean values for each group of questions.

Group 2 Requirements Quality	Group 3 Requirement influence in time/quality	Group 4 KT/KS	Group 5 Requirement quality re test
2	2	3	3
3	2	3	2
2	3	3	2
2	2	3	2
4	4	4	4
2	2	3	3
3	1	4	3
2	1	3	2
3	3	3	3
4	1	3	3
4	4	4	4

Source: Self-elaborated based on data gathered.

Given the means above, Pearson's correlation index was used to see relationships between each groups, Table 31 depicts the index calculations.

Table 31. Pearson's index between groups 2 to 5.

Correlation between groups	Pearson's correlation index
2 and 5	0.706699344
2 and 4	0.623609564
2 and 3	0.367597513
5 and 3	0.548556999
5 and 4	0.725866186
4 and 3	0.42320737

Source: Self-elaborated based on data gathered.

4.5.2.1 Analysis of findings

This section will be focused in correlational analysis between groups defined above, and based on information from table Table 31. There is a general observation, due to there are few levels for the answers the correlation index is too sensitive, generally a correlation over 0.5 could be used with caution, meanwhile a correlation above 0.7 is said to be a strong relationship evidence.

4.5.2.2 Correlation between SR quality attributes

The purpose of this calculation is to serve as a consistent index for answers about SR quality attributes, since group 2 and 5 asks almost about the same. There was just a little change on understandability and changeability SR properties.

4.5.2.3 Correlation between SR quality attributes and KT

Stronger relationships (above 0.7) are found between SR and KT, this can indicate that the more KT intensive process, the better SR process done.
The mean for group 4 (KT) is about 3.27 it indicates that KT is performed in a quite good level at organization ABC.

4.5.2.4 Requirements influence in time/quality lack of correlation

According to data gathered it seems that requirements per se doesn't influences the software project time or software quality. Was found that SR is not totally related with delay or errors in software projects. 8 of 10 of respondents disagree that SR per se at a cause for software projects delays and errors. This is correlated with understandability and validability, which occurs some times, giving room for new validation to be done causing delay and failures until final requirement is refined and implemented. This detail could be seen in Annex A, at section Req influence in time/quality.

4.6 Proposal Methodology feedback

Regarding the ***instrument*** respondents says: "Our process dictates a "knowledge transfer project" that is managed. This ensures that new person is fully aware of the processes involved, the tools involved and the standards involved. Items such as naming convention, abbreviations en terminology is also addressed. On the other side such transfers programs sometimes highlight shortfalls in a project that can be used to correct the missing or erroneous steps".

"Allow time for it (KT); often, in an agile environment, the sprint goals seem too high level, so the user stories (which act as the requirements) are incomplete, or superficial".

Most respondents says that motivation from the source and receipt were fundamental (41.67%) meanwhile other factors are divided as shown in Table 32. That was a surprise since motivation could sound obvious, and the same way the usage of a methodology.

Table 32. Respondents responsiveness about KT in SR.

How KT improve SR	Percentage of answers
Abstraction.	16.67%
Methodology.	16.67%
Motivation.	41.67%
Time access availability.	16.67%
Understandability.	8.33%

Source: Self-elaborated based on data gathered.

Regarding the process, respondents states that a method for requirements gathering should be mentioned and encourage for user availability.

Regarding the dimensions and aspects there were no feedback, maybe because SWEBOKv3 is extensive in mentioning SR stages.

Its encouraged that further correlational analysis should include ANOVA and t-test to complement the Pearson's index.

4.7 Chapter conclusion

This chapter attain this thesis objective 4, methodology testing, since application of the methodology was done; survey was constructed and executed based on framework defined and kickoff indicators refinement.

Internal consistency and correlational indicators were calculated. According to data gathered KT is correlated with SR.

According to data gathered it seems that requirements per se does not influences the software project time or software quality.

Motivation resulted as the most influence factor for SR performance, and curiously is done at the beginning of the KT process, and is not mentioned in classic SR methodologies.

5. Conclusions and recommendations

5.1 Research definition and research overview

A literature review was conducted in order to establish what is known about the process of knowledge transfer inside the software requirements stages. The literature review results in a lack of measurement tools and methodologies for software requirements and almost none for knowledge transfer.

In the existent literature about KT in software engineering, the principal focus has been:

- KT among software development multinationals.
- KT within projects in the organizations.
- KT between people within an organization.
- KT inside development teams, using agile models

Indicators for KT-SR measurement where gathered, classified and analyzed from the KT perspective. In order to do that, SR four stages (elicitation, specification, analysis and validation) were mapped against KT four stages (initialization, implementation, ramp-up, integration).

Even where there were more than 130 indicators defined, there was a KT stage who remained without indicators, such stage was the first (initialization), so, indicators were defined for it. Thus, the full indicators framework was completed.

A methodology for KT measurement was build based on framework defined, such methodology includes a data gathering specification with a plan including statistical tools for data analysis.

Motivation resulted as the most influence factor for SR performance, and curiously is done at the beginning of the KT process, and is not mentioned in classic SR methodologies.

KT for SR is quite different from classical approach to KT, because transfer of a full body of knowledge is not intended, instead of that, needs who relate with some business knowledge are transferred.

Since KT for SR is a special case, classical metrics used for knowledge transfer should not be applied. For instance, metrics oriented to intellectual capital, number of people trained does not make sense for any SR stage.

From the Study Case data, it could be inferred that KT is correlated with SR quality.

The mapping between SR and KT, serve as an approximation of KT measurement because shows how SR take place in each KT dimension and which factors influence each match.

Measure KT is a problem, because not exist any clear model that allow a quantitative and/or qualitative approximation to KT, until now.

5.2 Contributions to the body of knowledge

Two publications were done at the time this thesis was written. The first comprehends the state of the art (CAMACHO, SANCHEZ-TORRES, & GALVIS-LISTA, 2013) and the second was about the KT-SR mapping done (CAMACHO, SANCHEZ-TORRES, & GALVIS-LISTA, 2014).

5.3 Experimentation, Evaluation and Limitations.

A study case was performed in order to test the methodology proposed. A kickoff meeting was performed in order to choose indicators which best fits organization process. Next, a

survey was constructed based on indicators defined. Internal consistency was verified using Cronbach's alpha.

Data gathered was summarized in order to make correlational analysis using Pearson's coefficient.

Likert scale used could be a limitation due to we used just 4 level indicators, and maybe using 6 levels would result in more precise estimations. Likewise, another limitation is related with correlation, Spearman's rank or Kendall tau rank could be used in order to compare results.

5.4 Future work and research

Further work includes extending the survey to more people in order to compare results.

Next, include other software engineering process for KT consideration and measurement.

After that, a case tool in a wireframe like tool would be useful. Then, include other software engineering process for KT consideration and measurement.

Finally, a case tool in a wireframe like tool would be useful.

Bibliography

Abdullah, S., Selamat, M. H., Cob, Z. C., & Sazaly, U. S. (2011). A measurement framework for knowledge transfer in e-learning environment. *Information Technology Journal*, *10*(5), 927–943. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-79955923599&partnerID=40&md5=13c9b1714dd6ee7cf62ea42f7affa0e8

Abran, A. ., Al-Sarayreh, K. T. ., & Cuadrado-Gallego, J. J. . (2010). Measurement model of software requirements derived from system maintainability requirements. In *SEKE 2010 - Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering* (pp. 153–158). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-79952415890&partnerID=40&md5=d69497ee8001b0a4996030155c5bf220

Al-Sarayreh, K. T., Abran, A., & Santillo, L. (2010). Measurement of software requirements derived from system reliability requirements. In *24th European Conference on Object-Oriented Programming, ECOOP 2010 Workshop Proceedings - Workshop 1: Workshop on Advances in Functional Size Measurement and Effort Estimation, FSM'10*. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-79952391273&partnerID=40&md5=cc1a29ecc934bd348092a2ff931630eb

Alavi, M., & E. Leidner, D. (2001). Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues. *MIS Quarterly: Management Information Systems*, *25*(1), 107–136.

Albino, V. . b, Claudio Garavelli, A. . b c, & Schiuma, G. . c d e. (1998). Knowledge transfer and inter-firm relationships in industrial districts: The role of the leader firm. *Technovation*, *19*(1), 53–63. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0032786142&partnerID=40&md5=a9b6d6745539386e1cb7b9215015a3bd

Albino, V., Garavelli, A. C., & Gorgoglione, M. (2004). Organization and technology in knowledge transfer. *Benchmarking*, *11*(6), 584–600. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-23744512384&partnerID=40&md5=d222edacdb209fa33b53cefbff581e32

Ambos, T. C., & Ambos, B. (2009). The impact of distance on knowledge transfer effectiveness in multinational corporations. *Journal of International Management*, *15*(1), 1–14. doi:10.1016/j.intman.2008.02.002

Argote, L., & Ingram, P. (2000). Knowledge Transfer: A Basis for Competitive Advantage in Firms. *Organizational Behavior and Human Decision Processes*, *82*(1), 150–169. doi:10.1006/obhd.2000.2893

Aurum A., D. F. W. J. (2008). Investigating Knowledge Management practices in software development organisations - An Australian experience. *Information and Software Technology*, *50*(6), 511–533. doi:10.1016/j.infsof.2007.05.005

Avison, D., & Fitzgerald, G. (2006). *Information Systems Development*. (MCGraw-Hill., Ed.) (4th ed.). Berkshire: MCGraw-Hill.

Awad, M. A. (2005). *A comparison between agile and traditional software development methodologies*.

Banerjee, S. ., Sarkar, A. ., & Debnath, N. C. . (2013). Quality evaluation of requirement engineering framework: Business object based approach. In *2013 International Conference on Computing, Management and Telecommunications, ComManTel 2013* (pp. 374–379). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84875935074&partnerID=40&md5=5b2f28170e36a72f322df2bddc69ba98

Baruch Y.a Lin, C.-P. . (n.d.). All for one, one for all: Coopetition and virtual team performance. *Technological Forecasting and Social Change*. doi:10.1016/j.techfore.2012.01.008

Bjørnson F.O.a Dingsøyr, T. . (2005). A study of a mentoring program for knowledge transfer in a small software consultancy company. In K.-S. S. Bomarius F. (Ed.), *Lecture Notes in Computer Science* (Vol. 3547, pp. 245–256). Oulu. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-26444432231&partnerID=40&md5=0f6d99abd885a4e5e53f67f2d62d46f7

Boden, A., Avram, G., Bannon, L., & Wulf, V. (2009). Knowledge Management in Distributed Software Development Teams - Does Culture Matter? In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on* (pp. 18–27). doi:10.1109/ICGSE.2009.10

Borgatti, S. P., & Cross, R. (2003). A Relational View of Information Seeking and Learning in Social Networks. *Management Science*, *49*(4), 432–445.

CAMACHO, J. J., SANCHEZ-TORRES, J. M., & GALVIS-LISTA, E. (2013). Understanding the Process of Knowledge Transfer in Software Engineering : a Systematic Literature Review. *The International Journal of Soft Computing and Software Engineering*, *3*(3), 219–229. doi:10.7321/jscse.v3.n3.33

CAMACHO, J. J., SANCHEZ-TORRES, J. M., & GALVIS-LISTA, E. (2014). TOWARDS A KNOWLEDGE TRANSFER MEASUREMENT FOR SOFTWARE. In *International Conference on Information and Knowledge Engineering.* Las Vegas, NV. Retrieved from http://www.ucmss.com/cr/main/papersNew/papersAll/IKE3196.pdf

Carayannis, E. G. (1999). Knowledge transfer through technological hyperlearning in five industries. *Technovation*, *19*(3), 141–161.

Cares, C. ., & Franch, X. . (2009). Towards a framework for improving goal-oriented requirement models quality. In *Proceedings of the 12th Workshop on Requirements Engineering, WER 2009* (pp. 3–14). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84870402901&partnerID=40&md5=2ea6095d0cc579cff89a57d08dae9ebf

Chakraborty, S. ., Sarker, S. ., & Sarker, S. . (2010). An Exploration into the Process of Requirements. *Journal of the Association of Information Systems*, *11*(4), 212–249. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-77953090447&partnerID=40&md5=b5cf9913af4f28ed28a410b5eb088d07

Chau, T., Maurer, F., & Melnik, G. (2003). Knowledge sharing: agile methods vs. Tayloristic methods. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on* (pp. 302–307). doi:10.1109/ENABL.2003.1231427

Chen, C.-J. ., Shih, H.-A. ., & Yang, S.-Y. . (2009). The role of intellectual capital in knowledge transfer. *IEEE Transactions on Engineering Management*, *56*(3), 402–411. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-68249134122&partnerID=40&md5=cbd27dcc878d53c33281b70e06d82c99

Chen C.-J.a Shih, H.-A. . Y. S.-Y. . (2009). The role of intellectual capital in knowledge transfer. *IEEE Transactions on Engineering Management*, *56*(3), 402–411. doi:10.1109/TEM.2009.2023086

Chen, J.-S., & Lovvorn, A. S. (2011). The speed of knowledge transfer within multinational enterprises: the role of social capital. *International Journal of Commerce and Management*, *21*(1), 46–62. doi:10.1108/10569211111111694

Chen, Y.-J., Chen, Y.-M., & Chu, H.-C. (2008). Enabling collaborative product design through distributed engineering knowledge management. *Computers in Industry*, *59*(4), 395–409. doi:10.1016/j.compind.2007.10.001

Conejero, J. M. ., Figueiredo, E. ., Garcia, A. ., Hernández, J. ., & Jurado, E. . (2009). Early crosscutting metrics as predictors of software instability. *Lecture Notes in Business Information Processing*, *33 LNBIP*, 136–156. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-68949092270&partnerID=40&md5=84ab1a4c676b75d1b64d6242507b01eb

Conejero, J. M. ., Figueiredo, E. ., Garcia, A. ., Hernández, J. ., & Jurado, E. . (2012). On the relationship of concern metrics and requirements maintainability. *Information and Software Technology*, *54*(2), 212–238. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-81055140252&partnerID=40&md5=e93e4f19b29b4ca0b8d95616d915b154

Conradi, R. ., & Dybå, T. . (2001). An empirical study on the utility of formal routines to transfer knowledge and experience. In *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering* (pp. 268–276). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0035783712&partnerID=40&md5=bf52e51afcd637274470b62049763bf3

Costello, R. J. ., & Liu, D.-B. . (1995). Metrics for requirements engineering. *The Journal of Systems and Software*, *29*(1), 39–63. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0029292275&partnerID=40&md5=c1b9ebd7feb40e7ed8f7ab3d06c7f26a

Crowne, K. A. (2009). Enhancing knowledge transfer during and after international assignments. *Journal of Knowledge Management*, *13*(4), 134–147. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-70349671323&partnerID=40&md5=f5dc859b91cd44ccf5a12169f5744517

Damian D., M. S. K. I. (2007). Collaboration patterns and the impact of distance on awareness in requirements-centred social networks. In *Proceedings - 15th IEEE International Requirements Engineering Conference, RE 2007* (pp. 59–68). New Delhi. doi:10.1109/RE.2007.14

Damian, D., Marczak, S., & Kwan, I. (2007). Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centred Social Networks. In *Requirements Engineering Conference, 2007. RE '07. 15th IEEE International* (pp. 59–68). doi:10.1109/RE.2007.51

Dan, Z., Zhenqiang, B., Kaizhou, G., & Lei, G. (2008). Study on the Efficiency of Knowledge Transfer Based on Knowledge Transfer Scenario. In *Computer Science and Software Engineering, 2008 International Conference on* (Vol. 5, pp. 308–311). doi:10.1109/CSSE.2008.199

Davenport, B. T. H., Prusak, L., & Webber, A. (1998). Working Knowledge : How Organizations Manage What They Know. *Harvard Business School Press*, 1–15.

Desmarais, L. ., Parent, R. ., Leclerc, L. ., Raymond, L. ., Mackinnon, S. ., & Vézina, N. . (2009). Knowledge transfer between two geographically distant action research teams. *Journal of Workplace Learning*, *21*(3), 219–239. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-69949095114&partnerID=40&md5=7ce49a167753cd78d237e8f13c49f1d9

Dingsøyr, T., Bjørnson, F. O., & Shull, F. (2009). What Do We Know about Knowledge Management? Practical Implications for Software Engineering. *IEEE Software*, *26*(3), 100–103.

Duan, Y., Nie, W., & Coakes, E. (2010). Identifying key factors affecting transnational knowledge transfer. *Information & Management*, *47*(7-8), 356–363. doi:10.1016/j.im.2010.08.003

Espada, P., Goulão, M., & Araújo, J. (2011). Measuring complexity and completeness of KAOS goal models. In *Proceedings - 1st International Workshop on Empirical Requirements Engineering, EmpiRE 2011* (pp. 29–32). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-80455150297&partnerID=40&md5=cb769d655d52fb6c6f460ba8f8c66854

Farenhorst, R., & Vliet, H. Van. (2009). Understanding How to Support Architects in Sharing Knowledge. In *SHARK'09* (pp. 17–24).

Formentini, M., & Romano, P. (2011). Using value analysis to support knowledge transfer in the multi-project setting. *International Journal of Production Economics*, *131*(2), 545–560. doi:10.1016/j.ijpe.2011.01.023

Garavelli, C., Gorgoglione, M., & Scozzi, B. (2002). Managing knowledge transfer by knowledge technologies. *Technovation*, *22*, 269–279.

Gardner, P. L. ., Fong, A. Y. ., & Huang, R. L. . (2010). Measuring the impact of knowledge transfer from public research organisations: A comparison of metrics used around the world. *International Journal of Learning and Intellectual Capital*, *7*(3-4), 318–327. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-77955183838&partnerID=40&md5=3ace06cf7f3cdc4c62d6243124739ca8

Gardoni, M., Frank, C., & Vernadat, F. (2005). Knowledge capitalisation based on textual and graphical semi-structured and non-structured information: case study in an industrial research centre at EADS. *Computers in Industry*, *56*(1), 55–69. doi:10.1016/j.compind.2004.09.001

Génova, G. ., Fuentes, J. M. ., Llorens, J. ., Hurtado, O. ., & Moreno, V. . (2013). A framework to measure and improve the quality of textual requirements. *Requirements Engineering*, *18*(1), 25–41. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84874008264&partnerID=40&md5=05f27226b585872a2da73d060c8c59a8

Goh, D. H.-L., Chua, A. Y.-K., Luyt, B., & Lee, C. S. (2008). Knowledge access, creation and transfer in e-government portals. *Online Information Review*, *32*(3), 348–369. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-46249087586&partnerID=40&md5=6e4055f8071d4d2f7ddf55131d42d08e

Gorschek, T. ., & Davis, A. M. . (2008). Requirements engineering: In search of the dependent variables. *Information and Software Technology*, *50*(1-2), 67–75. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-36549080013&partnerID=40&md5=41e499461f677b895b241f1db31d7a10

Gosain, S. (2007). Mobilizing software expertise in personal knowledge exchanges ☆. *The Journal of Strategic Information Systems*, *16*(3), 254–277. doi:10.1016/j.jsis.2007.02.001

Hagge L., L. K. (2005). Sharing requirements engineering experience using patterns. *IEEE Software*, *22*(1), 24–31. doi:10.1109/MS.2005.17

Hagge, L., & Lappe, K. (2004). Patterns for the RE process. In *Proceedings of the IEEE International Conference on Requirements Engineering* (pp. 90–99). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-17044370455&partnerID=40&md5=9bb4caa634674af1a81e5b5eb40a7676

Hagge, L., & Lappe, K. (2005). Sharing requirements engineering experience using patterns. *Software, IEEE*, *22*(1), 24–31. doi:10.1109/MS.2005.17

Hanakawa, N. ., & Obana, M. . (2012). A metrics for meeting quality on a software requirement acquisition phase. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7343 LNCS*, 260–274. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84862205400&partnerID=40&md5=335bc8ca1d65053e8119eb1db8a5f622

Havlice, Z., Kunstar, J., Adamuscinova, I., & Plocica, O. (2009). Knowledge in software life cycle. *2009 7th International Symposium on Applied Machine Intelligence and Informatics*, 153–157. doi:10.1109/SAMI.2009.4956628

Holtsnider, B., Wheeler, T., Stragand, G., & Gee, J. (2010). Agile Development and Business Goals. In *Agile Development and Business Goal.* (pp. 11–29). Elsevier. doi:10.1016/B978-0-12-381520-0.00002-3

Hongmei, Q. ., & Huidong, W. . (2008). An empirical research on the influence of knowledge customization degree to knowledge transfer performance. In *2008 International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2008*. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-58049124024&partnerID=40&md5=2bd6f466aca4043620f0d3fa441ceb8c

Inkpen, A. C., & K. Tsang, E. W. (2005). SOCIAL CAPITAL , NETWORKS , AND KNOWLEDGE TRANSFER. *Academy of Management Review*, *30*(1), 146–165.

Iqbal, S., & Naeem Ahmed Khan, M. (2012). Yet another set of requirement metrics for software projects. *International Journal of Software Engineering and Its Applications*, *6*(1), 19–28. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84859841094&partnerID=40&md5=57001fb5e4aeba4d3ec1a54266f04600

Jasimuddin, S. M. (2007). Exploring knowledge transfer mechanisms: The case of a UK-based group within a high-tech global corporation. *International Journal of Information Management*, *27*(4), 294–300. doi:10.1016/j.ijinfomgt.2007.03.003

Jasimuddin, S. M. ., Connell, N. ., & Klein, J. H. . (2012). Knowledge transfer frameworks: An extension incorporating knowledge repositories and knowledge administration. *Information Systems Journal*, *22*(3), 195–209. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84860227089&partnerID=40&md5=3809fb284d746c02e6dd876b334f6065

Jørgensen, M., & Moløkken-Østvold, K. (2006). How large are software cost overruns? A review of the 1994 CHAOS report. *Information and Software Technology*, *48*(4), 297–301. doi:10.1016/j.infsof.2005.07.002

Karlsen, J. T., Hagman, L., & Pedersen, T. (2011). Intra-project transfer of knowledge in information systems development firms. *Journal of Systems and Information Technology*, *13*(1), 66–80. doi:10.1108/13287261111118359

Kedian, M. ., Zhi, J. ., & Didar, Z. . (2008). A Measurement-Driven process model for managing inconsistent software requirements. *Neonatal, Paediatric and Child Health*

Nursing, 291–298. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-60849130711&partnerID=40&md5=e7c5d2804c80cbf8facd438a9169945b

Khan, H., Ahmad, A., & Alnuem, M. A. (2012). Knowledge Management : A Solution to Requirements Understanding in Global Software Engineering. *Research Journal of Applied Sciences, Engineering and Technology*, *4*(14), 2087–2099.

Kitchenham, B. A. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering*.

Kjærgaard, A., Nielsen, P. A., & Kautz, K. (2010). Making Sense of Software Project Management A case of knowledge sharing in software development. *Scandinavian Journal of Information Systems*, *22*(1), 3–26.

Klendauer, R. ., Berkovich, M. ., Gelvin, R. ., Leimeister, J. M. ., & Krcmar, H. . (2012). Towards a competency model for requirements analysts. *Information Systems Journal*, *22*(6), 475–503. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84867741373&partnerID=40&md5=8566c55b2eb285a82ed5468e406432df

Ko, D.-G. ., Kirsch, L. J. ., & King, W. R. . (2005). Antecedents of knowledge transfer from consultants to clients in enterprise system implementations. *MIS Quarterly: Management Information Systems*, *29*(1), 59–85. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-33645514045&partnerID=40&md5=be2091196e98a50b0d1bf04f11c2cd9e

Koskinen, K. U., Pihlanto, P., & Vanharanta, H. (2003). Tacit knowledge acquisition and sharing in a project work context. *International Journal of Project Management*, *21*(4), 281–290. doi:10.1016/S0263-7863(02)00030-3

Krogh, G. Von. (2003). Understanding the problem of knowledge sharing. *International Journal of Information Technology and Management*, *2*(3), 173–183.

Krogh, G. Von, Nonaka, I., & Aben, M. (2001). Making the Most of Your Company ' s Knowledge : A Strategic Framework. *Long Range Planning*, *34*, 421–439.

Kuk, G. (2006). Strategic interaction and knowledge sharing in the KDE developer mailing list. *Management Science*, *52*(7), 1031–1042. doi:10.1287/mnsc.1060.0551

Kumar, J. A., & Ganesh, L. S. (2009). Research on knowledge transfer in organizations: a morphology. *Journal of Knowledge Management*, *13*(4), 161–174. doi:10.1108/13673270910971905

Kyaw P., B. C. R. S. (2003). A design recording framework to facilitate knowledge sharing in collaborative software engineering. In C. W. Chu W. (Ed.), *Proceedings of the IASTED International Conference on Information and Knowledge Sharing* (pp. 24–32). Scottdale, AZ. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-1542538740&partnerID=40&md5=db79651451a6ecbe0c7c71235cc026ba

Lam, W., Loomes, M., & Shankararaman, V. (1999). Managing requirements change using metrics and action planning. In *Proceedings of the Euromicro Conference on Software Maintenance and Reengineering, CSMR* (pp. 122–128). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0032665479&partnerID=40&md5=8fdd8a9a630271c6a8dcf4a9629f875a

Larman, C., & Basili, V. R. (2003). Iterative and Incremental Development : A brief History. *Computer*, *36*(6), 47–56.

Lee, S. B., & Shiva, S. G. (2009). A Novel Approach to Knowledge Sharing in Software Systems Engineering. In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on* (pp. 376–381). doi:10.1109/ICGSE.2009.59

Levine, D. I., & Gilbert, A. (1998). Knowledge Transfer: Managerial Practices Underlying One Piece of the Learning Organization. Berkeley: Center for Organization and Human Resource Effectiveness.

Li J.a Moe, N. B. . D. T. . (2010). Transition from a plan-driven process to Scrum - A longitudinal case study on software quality. In *ESEM 2010 - Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. Bolzano-Bozen. doi:10.1145/1852786.1852804

Liebowitz, J., & Suen, C. Y. (2000). Developing knowledge management metrics for measuring intellectual capital. *Journal of Intellectual Capital*, *1*(1).

Marshall, N., & Brady, T. (2001). Knowledge management and the politics of knowledge: illustrations from complex products and systems. *European Journal of Information Systems*, *10*(2), 99–112. doi:10.1057/palgrave.ejis.3000398

Mavin, A., & Wilkinson, P. (2010). BIG EARS (The return of “Easy Approach to Requirements Syntax”). In *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference, RE2010* (pp. 277–282). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-78650371993&partnerID=40&md5=7bb08d508448f3bcd72022ee850377ee

McGee, S., & Greer, D. (2012). Towards an understanding of the causes and effects of software requirements change: Two case studies. *Requirements Engineering*, *17*(2), 133–155. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84861702910&partnerID=40&md5=4e836a0dd3ea3a7b913299beb7fcc161

Mei, Y., Wang, Z., & Cao, Z. (2011). Performance evaluation model of knowledge transfer. In *2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce, AIMSEC 2011 - Proceedings* (pp. 5677–5681). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-80053232753&partnerID=40&md5=b12ea03396076c708e1cb86a6419c06d

Mestad, A., Myrdal, R., Dingsoyr, T., & Dyba, T. (2007). Building a Learning Organization: Three Phases of Communities of Practice in a Software Consulting Company. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on* (p. 189a). doi:10.1109/HICSS.2007.115

Mestad A.a Myrdal, R. . D. T. . D. T. . (2007). Building a learning organization: Three phases of communities of practice in a software consulting company. In *Proceedings of the Annual Hawaii International Conference on System Sciences*. Big Island, HI. doi:10.1109/HICSS.2007.115

Minbaeva, D. B. (2007). Knowledge transfer in multinational corporations. *Management International Review*, *47*(4), 567–593. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-34548201654&partnerID=40&md5=cd3b011dfd9b6b752aeb163d4a33e0d2

Mora, M. (2011). Validity and Reliability in Surveys. Retrieved April 04, 2014, from http://www.relevantinsights.com/validity-and-reliability

Niazi, M. ., Cox, K. ., & Verner, J. . (2008). A measurement framework for assessing the maturity of requirements engineering process. *Software Quality Journal*, *16*(2), 213–235. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-42149166093&partnerID=40&md5=59d798babdfb2cc37443881fb141ca6b

Niederman, F. (2005). International business and MIS approaches to multinational organizational research: The cases of knowledge transfer and IT workforce outsourcing. *Journal of International Management*, *11*(2), 187–200. doi:10.1016/j.intman.2005.03.004

Nonaka, I. (2007). The Knowledge-Creating Company. *Harvard Business Review*, *85*(August), 162–194.

Nonaka, I., & Toyama, R. (2003). The knowledge-creating theory revisited: knowledge creation as a synthesizing process. *Knowledge Management Research & Practice*, *1*(1), 2–10. doi:10.1057/palgrave.kmrp.8500001

Pilat, L., & Kaindl, H. (2011). A knowledge management perspective of requirements engineering. In *Proceedings - International Conference on Research Challenges in Information Science*. Gosier. doi:10.1109/RCIS.2011.6006849

Poort E.R.a Pramono, A. . P. M. . C. V. . V. V. H. . (2009). Successful architectural knowledge sharing: Beware of emotions. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5581 LNCS*, 130–145. doi:10.1007/978-3-642-02351-4_9

Qu G.a Ji, S. . N. A. . (2011). Project complexity and knowledge transfer in global software outsourcing project teams: A transactive memory systems perspective. In *Proceedings of the Annual Hawaii International Conference on System Sciences* (pp. 3776–3785). Maui, HI. doi:10.1109/HICSS.2012.488

Ras, E. (2009). Investigating wikis for software engineering - Results of two case studies. In *Proceedings - International Conference on Software Engineering* (pp. 47–55). Vancouver, BC. doi:10.1109/WIKIS4SE.2009.5069996

Ras, E., Gabriela, A., Patrick, W., & Stephan, W. (2005). Using weblogs for knowledge sharing and learning in information spaces. *Journal of Universal Computer Science*, *11*(3), 394–409. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-23844525369&partnerID=40&md5=cdf909ce70ad49d5f9f4f580e01bd397

Ren, T. (2009). The evaluation model of knowledge worker's knowledge-sharing performance. In *2009 International Conference on Web Information Systems and Mining, WISM 2009* (pp. 819–822). Shanghai. doi:10.1109/WISM.2009.170

Rezgui, Y., Hopfe, C. J., & Vorakulpipat, C. (2010). Generations of knowledge management in the architecture, engineering and construction industry: An evolutionary perspective. *Advanced Engineering Informatics*, *24*(2), 219–228. doi:10.1016/j.aei.2009.12.001

Roy, S. Sen. (2011). *EXPLORING THE PROPENSITY TO SHARE PRODUCT INFORMATION ON SOCIAL NETWORKS*. UNIVERSITY OF MINNESOTA.

Rus, I., & Lindvall, M. (2002). Knowledge management in software engineering. *IEEE Software*, *19*(3), 26–38. doi:10.1109/MS.2002.1003450

Salger, F., & Engels, G. (2010). Knowledge transfer in global software development: leveraging acceptance test case specifications. In *Software Engineering, 2010 ACM/IEEE 32nd International Conference on* (Vol. 2, pp. 211–214). doi:10.1145/1810295.1810332

Salger, F., Sauer, S., Engels, G., & Baumann, A. (2010). Knowledge Transfer in Global Software Development - Leveraging Ontologies, Tools and Assessments. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on* (pp. 336–341). doi:10.1109/ICGSE.2010.46

Salger F.a Engels, G. . (2010). Knowledge transfer in global software development - Leveraging acceptance test case specifications. In *Proceedings - International Conference on Software Engineering* (Vol. 2, pp. 211–214). Cape Town. doi:10.1145/1810295.1810332

Sánchez-torres, J. M., Carolina, S., & Torres, R. (2009). A model for measuring research capacity using an intellectual capital-based approach in a colombian higher education institution Do HEIs ' contributions represent capacities in the Other questions have risen in turn : *Innovar [online]*, *19*(1), 179–197. Retrieved from http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0121-50512009000400013&lng=en&nrm=iso

Schneider, K. (2009). *Experience and Knowledge Management in Software Engineering* (pp. 165–202). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-540-95880-2

Schwartz, D. G. (2007). Integrating knowledge transfer and computer-mediated communication: Categorizing barriers and possible responses. *Knowledge Management Research and Practice*, *5*(4), 249–259. Retrieved from

http://www.scopus.com/inward/record.url?eid=2-s2.0-36148975356&partnerID=40&md5=00d72036f5bdbce05602384a116875d1

Shan, X., Jiang, G., & Huang, T. (2010). The study on knowledge transfer of software project requirements. In *2010 International Conference on Biomedical Engineering and Computer Science, ICBECS 2010*. Wuhan. doi:10.1109/ICBECS.2010.5462314

Shan, X., Jiang, G., & Huang, T. (2010). The study on knowledge transfer of software project requirements. In *2010 International Conference on Biomedical Engineering and Computer Science, ICBECS 2010*. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-77953314490&partnerID=40&md5=3ab3f5064adf49083cdf18e4591297cd

Shou Y., S. Y. (2010). Modeling and simulation of knowledge transfer within an inter-firm network. In *Proceedings - 2010 International Conference of Information Science and Management Engineering, ISME 2010* (Vol. 2, pp. 99–103). Xi'an. doi:10.1109/ISME.2010.137

Shou, Y., & Sun, Y. (2010). Modeling and simulation of knowledge transfer within an inter-firm network. In *Proceedings - 2010 International Conference of Information Science and Management Engineering, ISME 2010* (Vol. 2, pp. 99–103). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-78049345965&partnerID=40&md5=db65ccbb87b80111569679836b89a06e

Simonin, B. L. . b c d e f g. (2004). An empirical investigation of the process of knowledge transfer in international strategic alliances. *Journal of International Business Studies*, *35*(5), 407–427. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-7444254131&partnerID=40&md5=092408754467ce9144d1bbc0b9e1f7da

Szulanski, G. (1996). Exploring internal stickiness: Impediments to the transfer of best practice within the firm. *Strategic Management Journal*, *17*(SUPPL. WINTER), 27–43. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0041751098&partnerID=40&md5=311433b07b40d218693f39a5fe3897dd

Szulanski, G. (2000). The Process of Knowledge Transfer: A Diachronic Analysis of Stickiness. *Organizational Behavior and Human Decision Processes*, *82*(1), 9–27. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0001311775&partnerID=40&md5=f56a49022a9c01291b88a446d0f00c4a

Tesch, D., Sobol, M. G., Klein, G., & Jiang, J. J. (2009). User and developer common knowledge: Effect on the success of information system development projects. *International Journal of Project Management*, *27*(7), 657–664. doi:10.1016/j.ijproman.2009.01.002

Verkasalo, M. . c, & Lappalainen, P. . (1998). A method of measuring the efficiency of the knowledge utilization process. *IEEE Transactions on Engineering Management*, *45*(4), 414–423. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0032204893&partnerID=40&md5=1616e85f412b0202ef9f51a8167a6673

Wah C.Y., M. T. L. B. E. H.-D. (2005). Theorizing, measuring, and predicting knowledge sharing behavior in organizations - A social capital approach. In J. R. H. Spraque (Ed.), *Proceedings of the Annual Hawaii International Conference on System Sciences* (p. 252). Big Island, HI. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-27544509659&partnerID=40&md5=29e9c3918e5cbd2ee1bb6735d0fb15be

Walonick, D. S. (2012). Survey Design Guidelines. Retrieved April 04, 2014, from http://www.statpac.com/survey-design-guidelines.htm

Wang, J.-R., & Yang, J. (2008). Study on Knowledge Sharing Behavior in Software Development Team. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on* (pp. 1–4). doi:10.1109/WiCom.2008.2951

Ward, J., & Aurum, A. (2004). Knowledge management in software engineering - describing the process. *2004 Australian Software Engineering Conference. Proceedings.*, (c), 137–146. doi:10.1109/ASWEC.2004.1290466

Watson, S., & Hewett, K. (2006). A multi-theoretical model of knowledge transfer in organizations: Determinants of knowledge contribution and knowledge reuse. *Journal of Management Studies*, *43*(2), 141–173. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-33645135147&partnerID=40&md5=104a7def37f1ed263758ca09937d22b4

Whitworth, E. (2006). Introduction to agile software development .pdf.

Wilkesmann, M., & Wilkesmann, U. (2011). Knowledge transfer as interaction between experts and novices supported by technology. *Vine*, *41*(2), 96–112. doi:10.1108/03055721111134763

Windiarti, I. S., Ferris, T. L. J., & Berryman, M. J. (2011). Technology and knowledge sharing strategy in systems engineering practice performed by Indonesian expatriate engineers. In *Industrial Engineering and Engineering Management (IEEM), 2011 IEEE International Conference on* (pp. 509–513). doi:10.1109/IEEM.2011.6117969

Wiradanti, B., & Govindaraju, R. (2011). Requirements engineering maturity measurement and evaluation (A case study of Bank X in Indonesia). In *2011 IEEE 3rd International Conference on Communication Software and Networks, ICCSN 2011* (pp. 164–169). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-80053169068&partnerID=40&md5=4de09b363b5e5e4acb818e0e7f0ed62e

Yavari, Y. ., Afsharchi, M. ., & Karami, M. . (2011). Software complexity level determination using software effort estimation use case points metrics. In *2011 5th Malaysian Conference in Software Engineering, MySEC 2011* (pp. 257–262). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84857261131&partnerID=40&md5=6c630f1db916e0dbb73d226d8effaafb

Zhang, D., Bao, Z., Gao, K., & Guo, L. (2008). Study on the efficiency of knowledge transfer based on knowledge transfer scenario. In *Proceedings - International*

Conference on Computer Science and Software Engineering, CSSE 2008 (Vol. 5, pp. 308–311). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-79951483814&partnerID=40&md5=1245a5d98d8c5606a7bd218a6bc8006b

Zhang Q., D. R. (2011). Impacts of cultural difference on knowledge sharing, relationship quality and performance in IT-based service outsourcing. In *2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce, AIMSEC 2011 - Proceedings* (pp. 6271–6274). Zhengzhou. doi:10.1109/AIMSEC.2011.6011441

Appendix A: Responses gathered.

Series of questions used to measure each KT-SR mapping (scope) ordered by analysis dimensions. Answers for groups 2 to 5 are displayed using Table 33, Table 34, Table 35 and

Table 36.

Table 33. Group 2 answers.

Requirements quality attributes							
How many actors, extension points or dependencies, does a requirement have?	[There is just one single interpretation for each requirement.]	[Each requirement is clearly determined and identified, overlapping among requirements.]	[Terms used in requirements are concrete and well defined.]	[All requirements and their dependencies are specified.]	[There are compliance monitoring for requirements, at every stage in software life-cycle.]	[Users state that requirements express their needs effectively.]	[Software produced fulfills requirements specification.]
1	3	2	2	2	2	3	3
2	2	3	4	4	3	4	3
1	1	2	3	3	1	3	3
2	3	1	2	1	3	3	3
2	4	4	4	4	4	4	4
2	3	2	2	3	1	1	3
2	1	2	3	2	4	4	4
1	1	2	3	1	1	2	3
3	2	2	2	2	2	2	2
2	2	2	2	2	1	1	3
1	3	3	4	4	4	3	2
19	25	25	31	28	26	30	33
k	6						
sum Var	5.950413223						
var	20						
Cronbach's alpha Req Quality Attributes	0.842975207						

Source: self-elaboration.

Table 34. Group 3 answers.

Requirements influence in time/quality	
[Requirements cause delay or failures in software projects.]	[Requirements may cause a poor software quality (stability, scalability, completeness, etc.)]
2	1
2	1
4	2
2	2
1	1
2	1
1	1
1	1
3	3
1	1
2	1
21	15
k	2
sum Var	1.223140496
var	2.016528926
conrbach alpha Req Quality Attributes	0.786885246

Source: self-elaboration.

Table 35. Group 4 answers.

Awareness / Itention / Motivation to share / transfer knowledge							
[I feel happy with my experiences, so I want others to know them.]	[I want to share my huge experience.]	[I want to help others with my positives experiences.]	[I want to save others from bad/negative experiences I've had.]	[I don't feel happy with the software process which is currently done.]	[I feel good when talking with others about my successful experiences.]	[I want to be rewarded for sharing my knowledge.]	[I like to talk with people with common interests.]
3	3	3	3	3	2	3	2
3	3	3	3	2	3	3	3
3	3	3	3	4	4	3	3
3	3	3	3	4	3	2	4
1	1	1	1	1	1	1	1
3	3	3	3	3	3	2	4
4	4	4	4	1	3	4	4
3	3	3	3	2	3	2	3
3	3	3	3	3	3	3	3
3	2	2	3	4	2	2	3
1	2	1	1	1	2	1	2
30	30	29	30	28	29	26	32
K	8						
sum Var	6.347107438						
Var	34.74380165						
Cronbach's alpha Req Quality Attributes	0.93407639						

Source: self-elaborated.

Table 36. Group 5 answers.

Perception about requirements changeability related properties					
[Are correctly understood, without difficulty.]	[Are specified without explain technical details about implementation.]	[Meets users needs.]	[Stop changing over time.]	[Change due to failures or errors in their specification.]	[Change because of user validations.]
3	3	3	2	3	3
3	2	4	1	2	2
3	2	3	1	2	2
2	2	2	1	2	3
4	4	4	4	4	4
2	4	2	1	4	4
2	3	4	3	2	3
2	2	3	1	2	4
2	3	2	2	3	3
2	3	3	2	3	3
4	4	3	3	3	4
29	32	33	21	30	35
K	5				
sum Var	3.289256198				
Var	9.867768595				
conrbach alpha Req Quality Attributes	0.833333333				

Source: self-elaborated

Appendix B: Full questionnaire report.

The next pages will show the results from the questionnaire applied. There are four questions groups (people profile, SR Quality Attributes, SR link with delay and errors, KT and SR quality attributes re test) as stated in sections 3.2.4 and 4.3 where data gathering were defined and applied.

- *People profile*

We can say that we have a mixed population since most of the respondents (63%) work in high sized companies, the other 37% works in small and medium sized companies. So, we expect a general view of SR and KT.

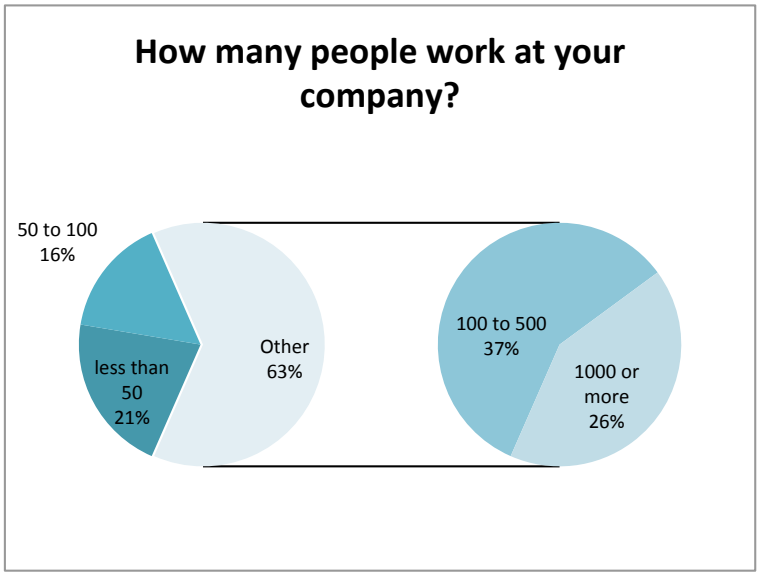


Figure 17. How many people work at your company?
Source: Self-elaborated.

Developers, software architects/tech leads and software analysts are the principal roles who characterize the respondents.

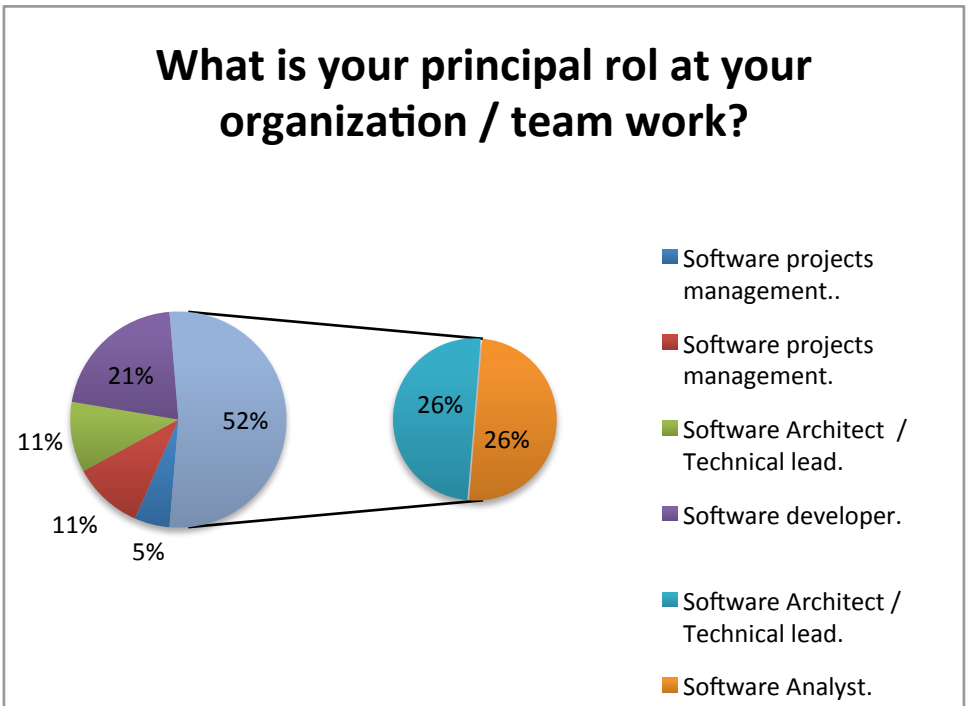


Figure 18. What's your principal rol at your organization / team work?
Source: Self-elaborated.

From answers gathered it could be seen that most of the software requirements teams are small, conformed whit less than 10 people.

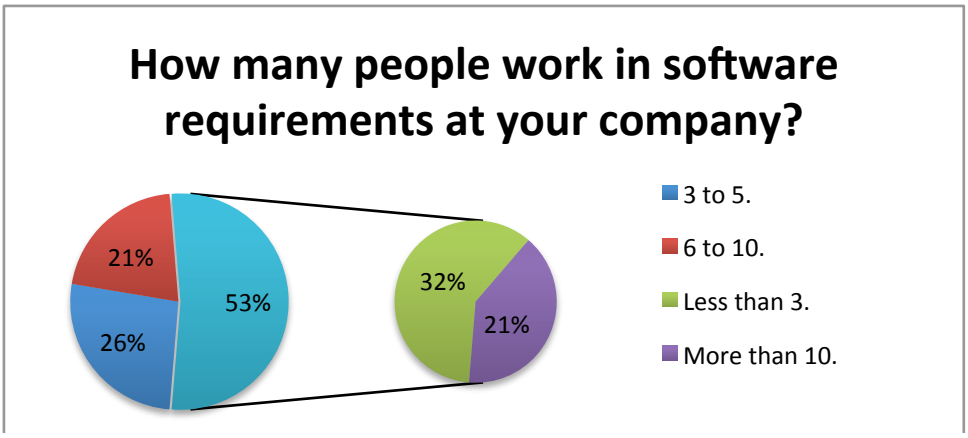


Figure 19. How many people work in software requirements at your company?
Source: self-elaborated.

74% of respondents have a significant experience from 4 to 7 or more year's experience. So we can expect answers coming from expert people.

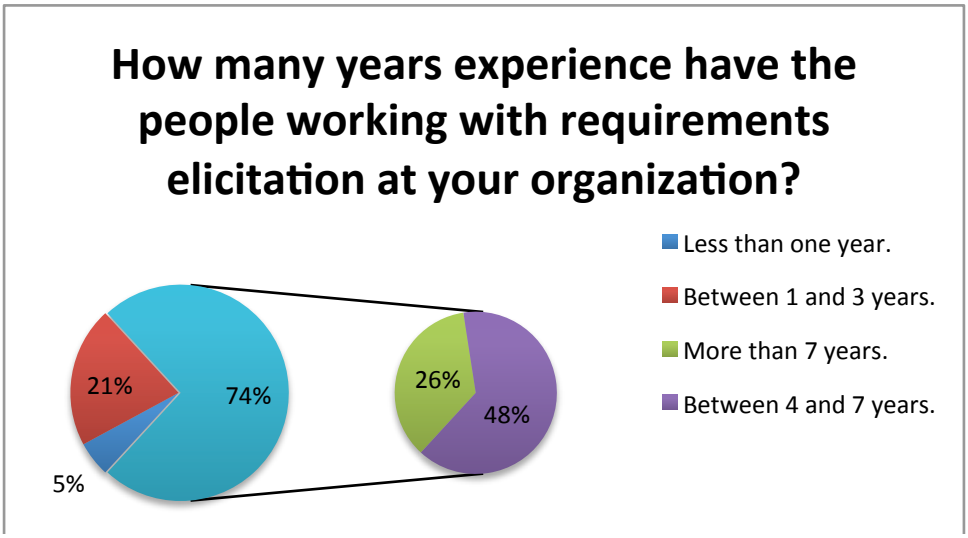


Figure 20. How many year's experience have the people working with requirements at your organization?
Source: Self-elaborated.

- *SR quality attributes*

According our complexity indicator defined in 3.2.2, 90% of requirements are low to mid complex. So one could say that complexity is controlled according respondents.

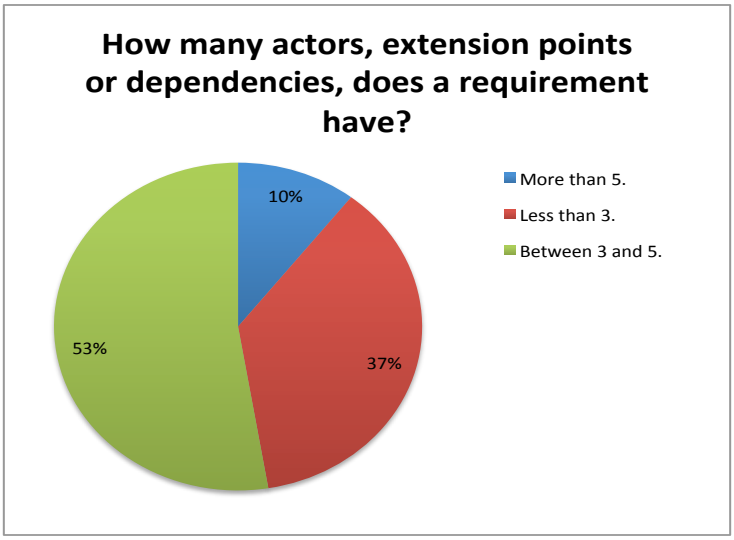


Figure 21. Complexity.
Source: Self-elaborated.

When concerning ambiguity, we found that there is a lot of ambiguity; most of respondents affirm that ambiguity is rarely avoided.

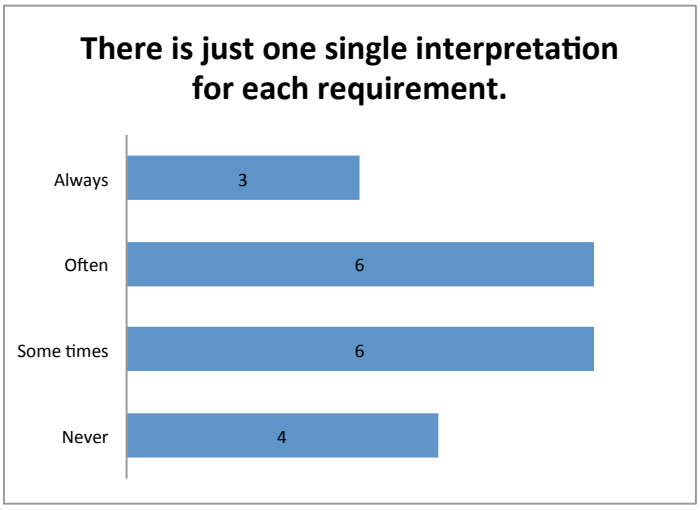


Figure 22. Ambiguity.
Source: Self-elaborated.

It was found that Atomicity equals than Ambiguity is rarely avoided.

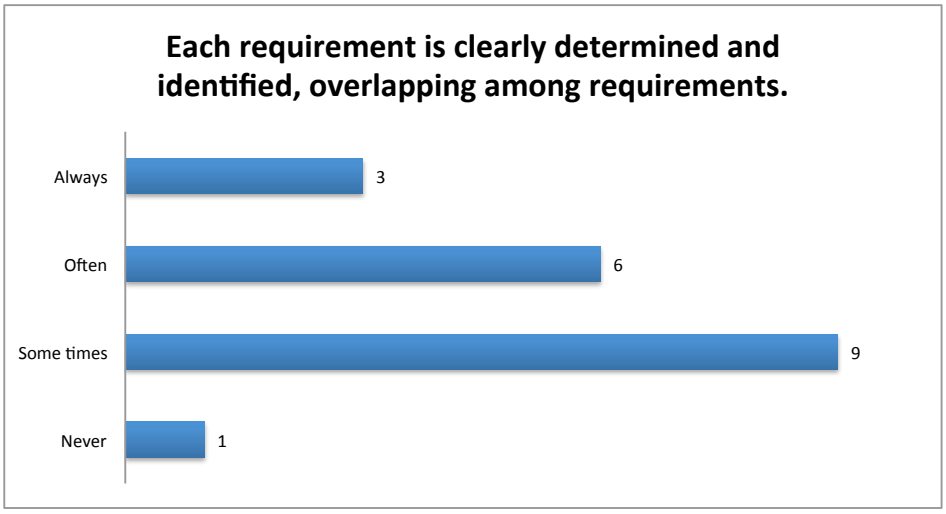


Figure 23. Atomicity.
Source: Self-elaborated.

It's interesting that even when ambiguity and atomicity are rarely achieved, the precision have better performance, it could be done since there are more preoccupation into build a software specification, even when it is not too accurate. Something good is that requirements are never without any mean of precision.

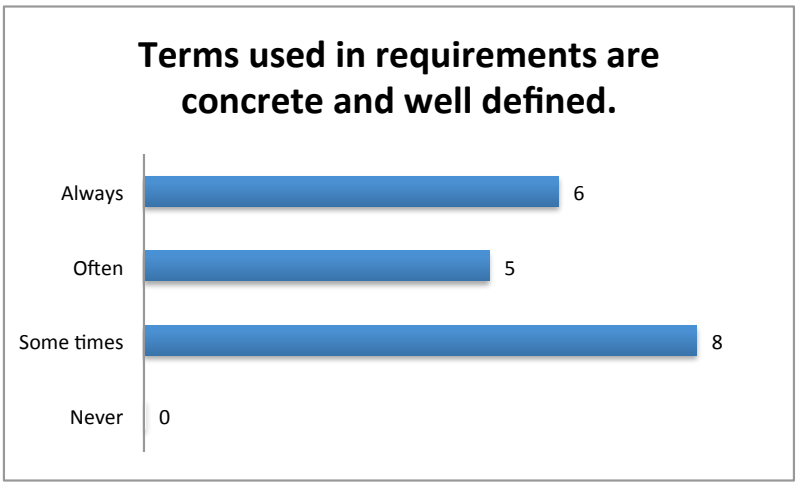


Figure 24. Precision.
Source: Self-elaborated.

About completeness, more than half respondents state that is never totally achieved, what is makes sense taking into account that previous indicators are rarely totally fulfilled.

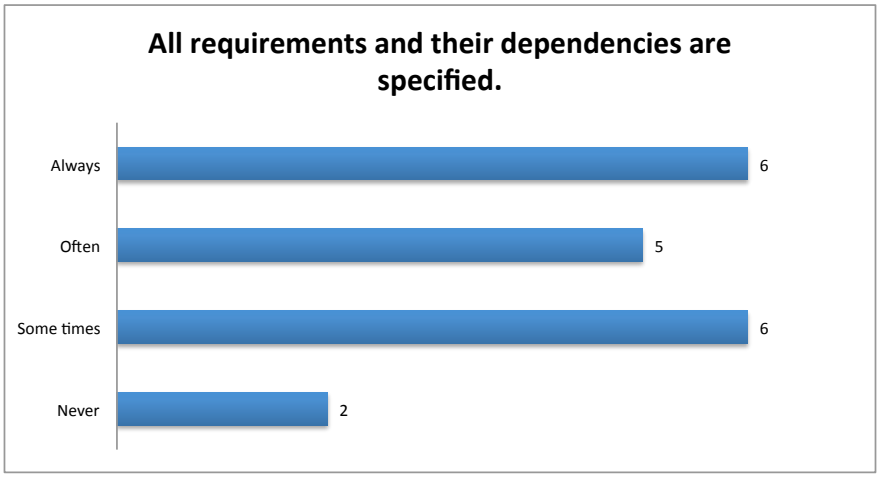


Figure 25. Completeness
Source: Self-elaborated.

Regarding traceability, the perception is the same than complexity and again makes sense since ambiguous, incomplete, low atomicity and low precision requirements are very hard to trace.

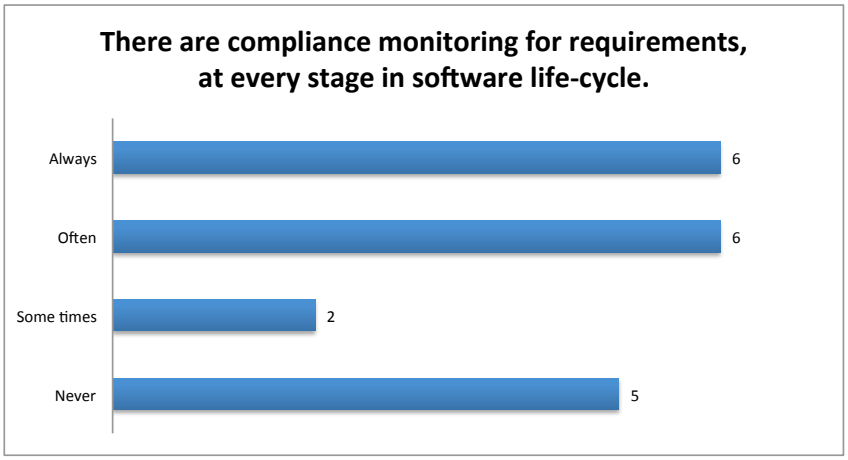


Figure 26. Traceability.
Source: Self-elaborated.

Considering validability, it's seen that it behave a little better than traceability. Very often requirements express what user really need, this shows that not always requirements express what client needs what is correlated with the high ambiguity and low precision as stated before.

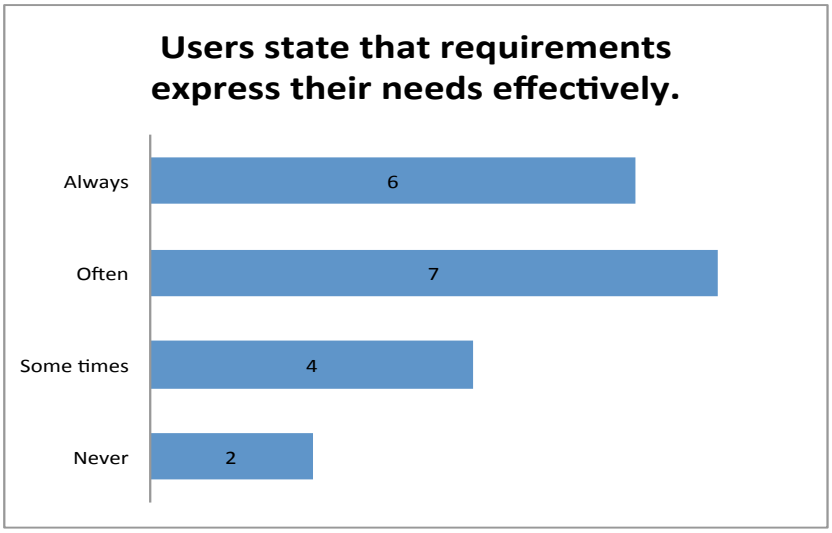


Figure 27. Validability.
Source: Self-elaborated.

The software produced often fulfills requirements specification, which is correlated with low performance of atomicity and precision as stated before. And, correlates with correctness indicator as stated in Figure 31.

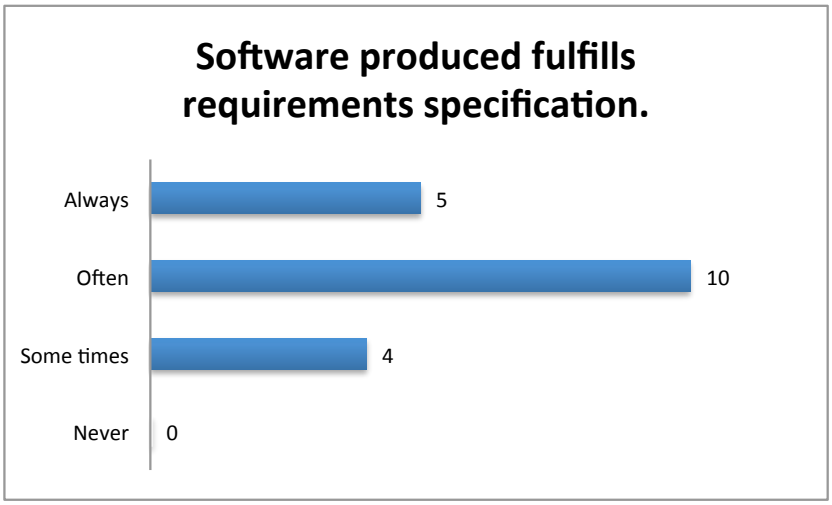


Figure 28. Software fulfillment.
Source: Self-elaborated.

- *SR quality attributes re test*

Understandability was found to be quite positive since never are totally not understood, but stills being some times the major frequency for understandability, which is correlated with low atomicity, low precision and ambiguity as stated before.

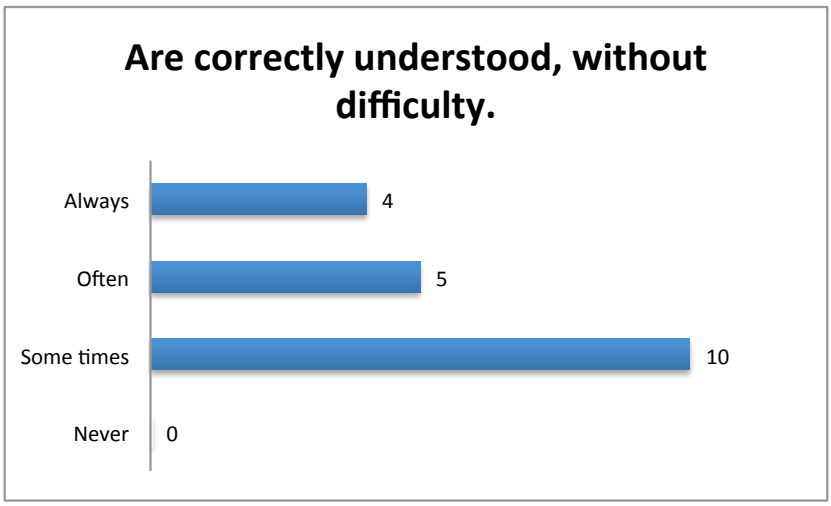


Figure 29. Understandability.
Source: self-elaborated.

Full Abstraction is never achieved, but more often it is done. It is correlated with atomicity and precision, if atomicity, precision and abstraction are low, and requirements are just some times understandable, technical details can be introduced, resulting in low abstraction.

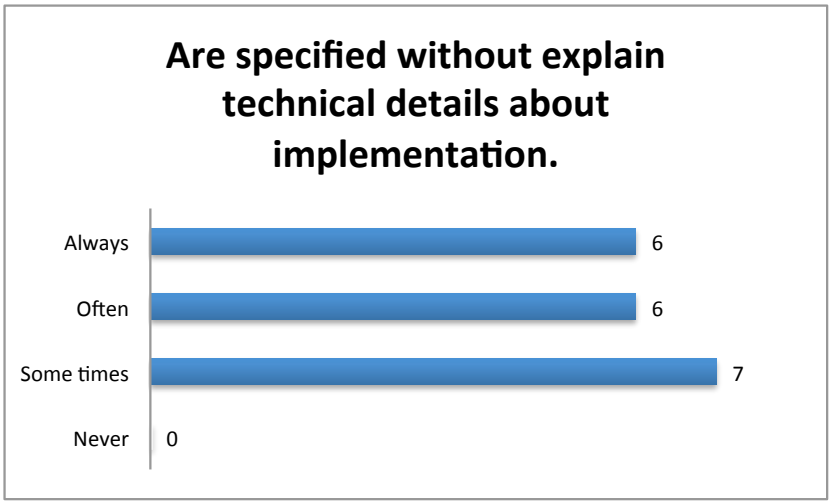


Figure 30. Abstraction.
Source: self-elaborated.

Correctness is often achieved, what is correlated with the validability which is very often achieved.

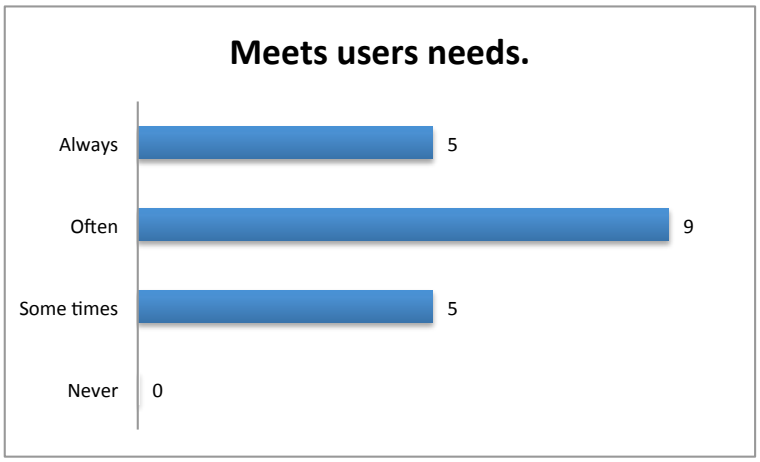


Figure 31. Correctness.
Source: self-elaborated.

Its clear that requirements never stop changing, what is related with previous poor indicators of ambiguity and precision.

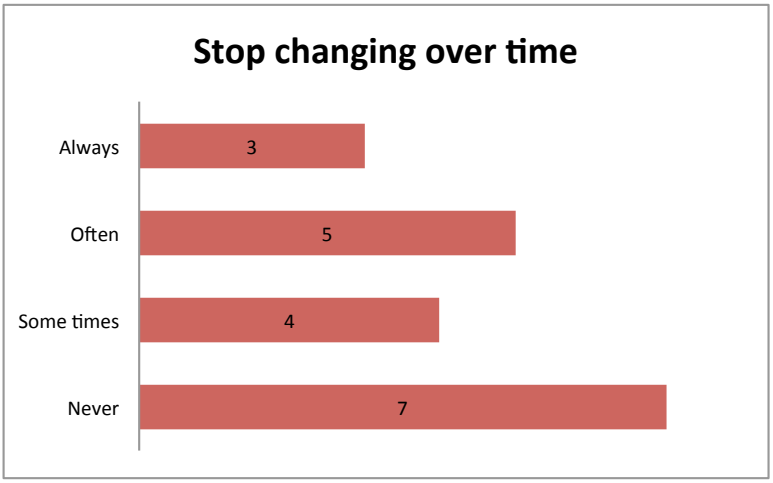


Figure 32. Changeability.
Source: self-elaborated.

Some times requirements change due to errors in their specification, which correlate with ambiguity, lack of precision and atomicity.

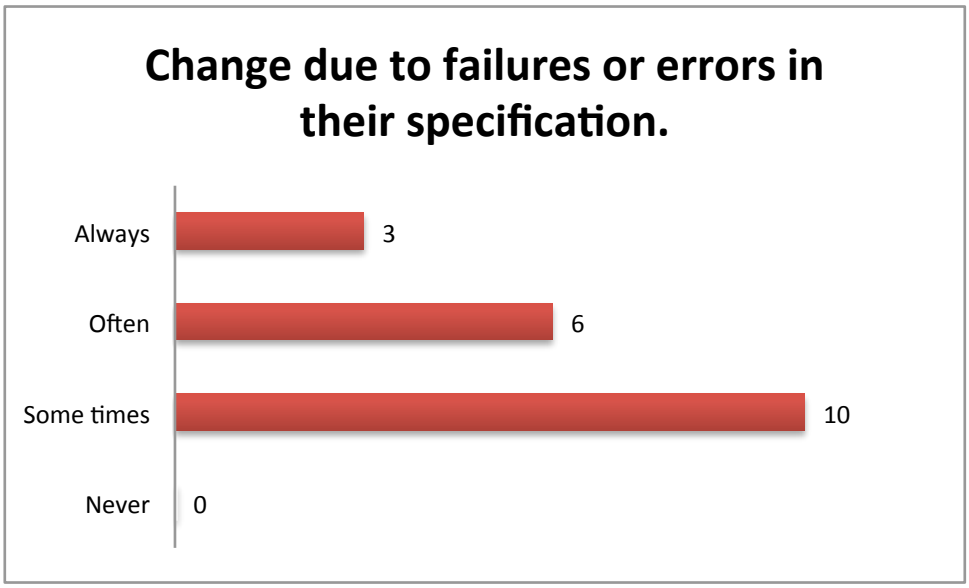


Figure 33. Changeability due to SRS.
Source: self-elaborated.

Changeability due to user validations occur some times, but in less quantity than due to errors in SRS, this make sense since good understandability happens some time as stated before, giving a lot of room for requirements changes at validation stage.

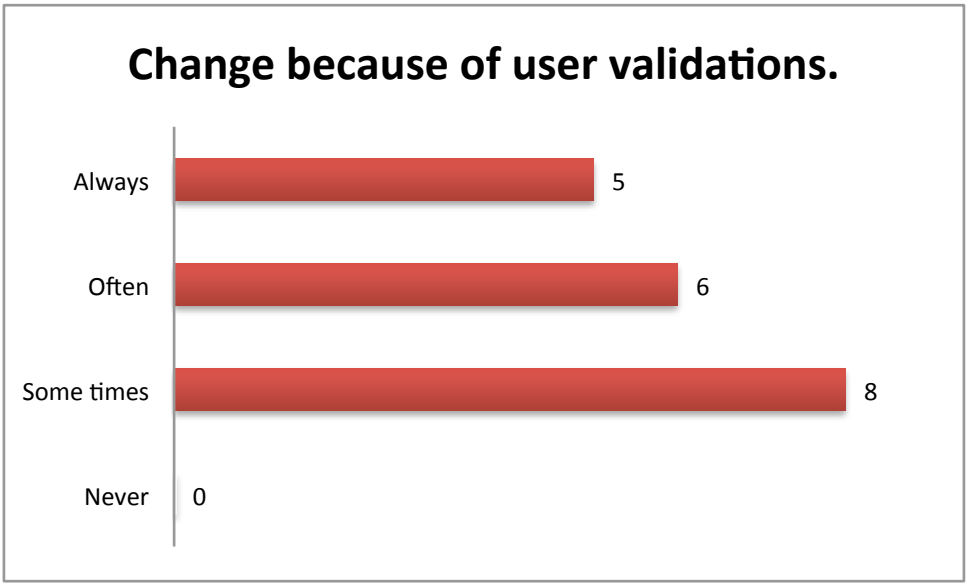


Figure 34. Changeability due to validation.
Source: self-elaborated.

- *SR link with delay and errors*

It was found that the SR are not totally related with delay or errors in software projects. Most of respondents disagree that SR per se at a cause for software projects delays and errors. This is correlated with understandability and validability which occurs some times, giving room for new validation to be done causing delay and failures until final requirement is refined and implemented.

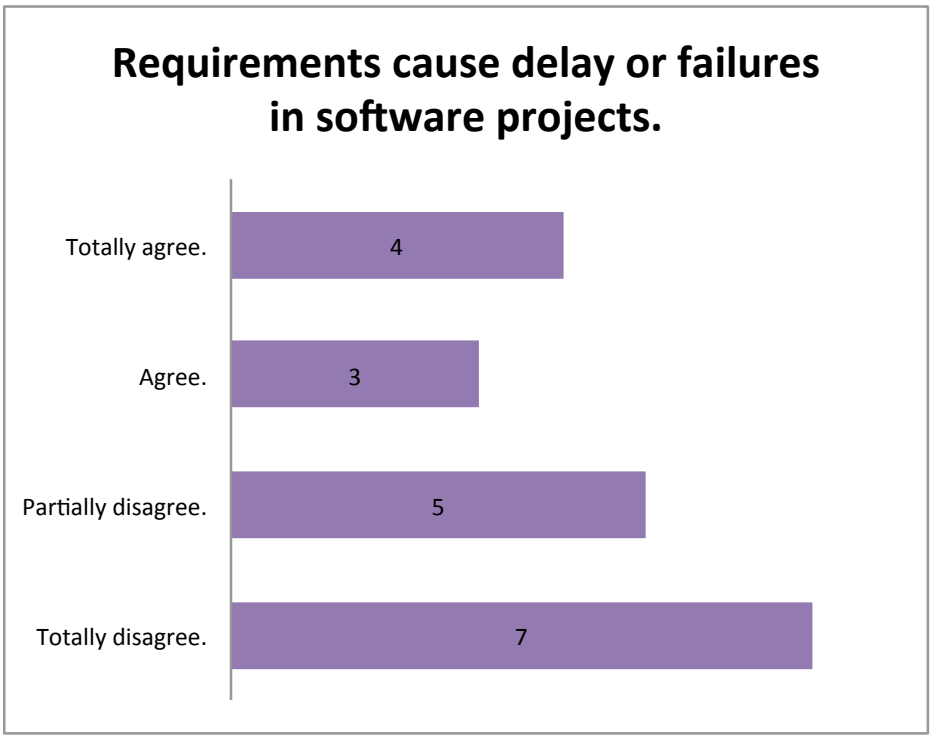


Figure 35. SR and delay or failures in software projects.
Source: self-elaborated.

A weak influence SR influence in other software quality attributes was found. So not possible correlation could be done with indicators gathered before.

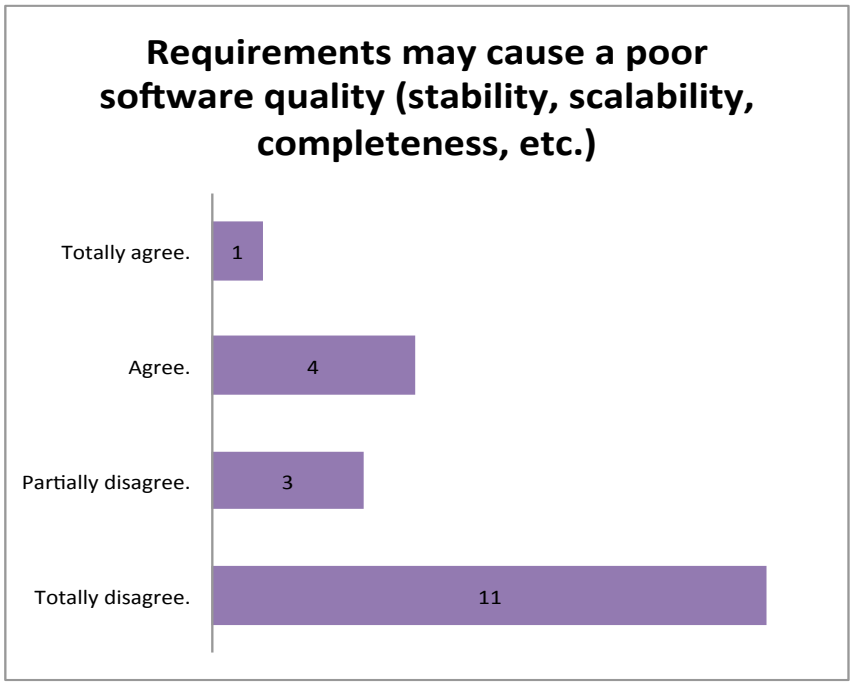


Figure 36. SR and software quality attributes.
Source: self-elaborated.

- *KT motivation*

A big agreement was found concerning intention to share.

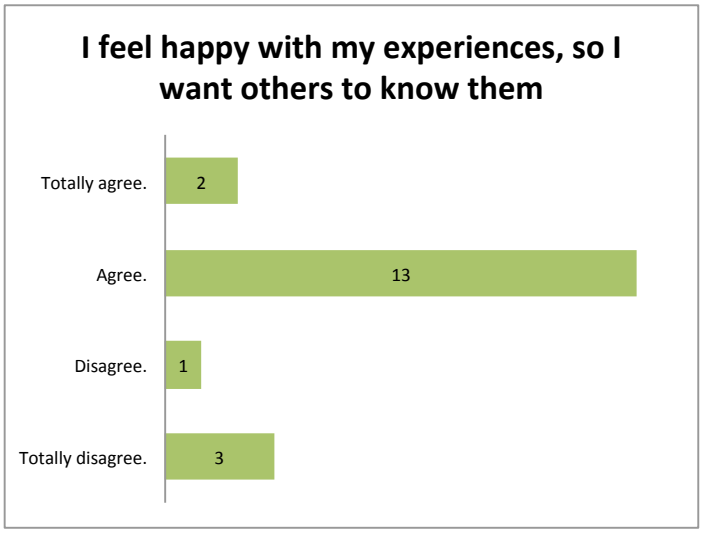


Figure 37. Intention to share happy experiences.
Source: self-elaborated.

Respondents agree about their intention to share their huge experiences.

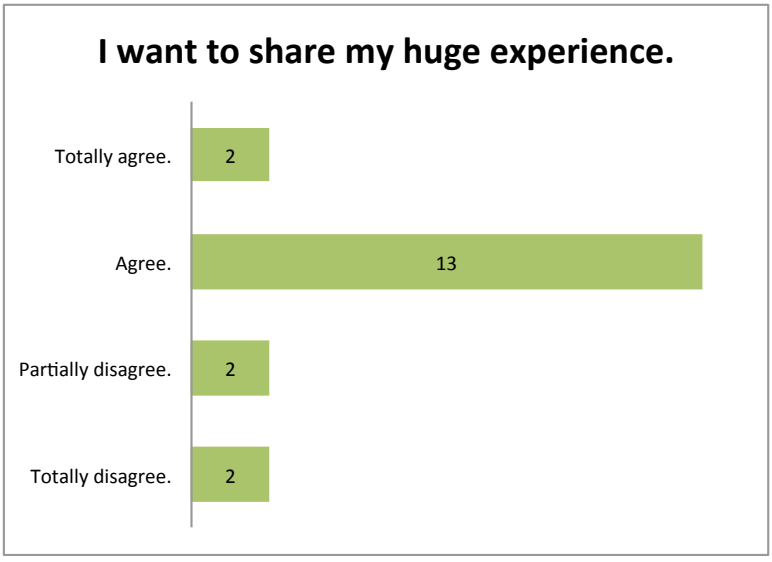


Figure 38. Intention to share huge experience.
Source: self-elaborated.

There is a general agreement about share positive experiences.

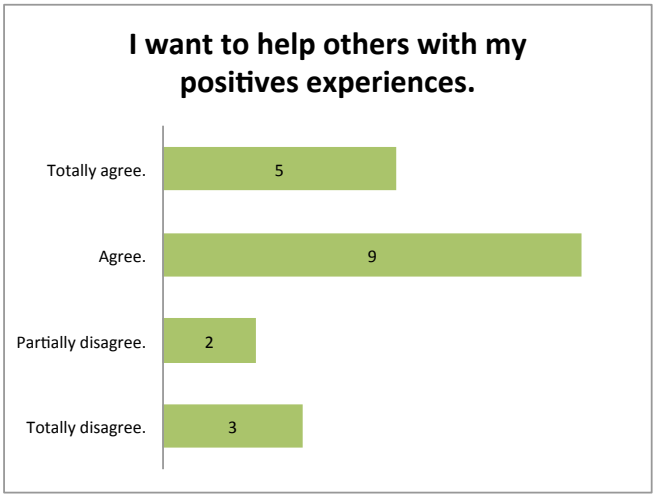


Figure 39. Intention to share positive experiences.
Source: self-elaborated.

There is a big agreement about the willingness to share knowledge in order to save others from pitfalls.

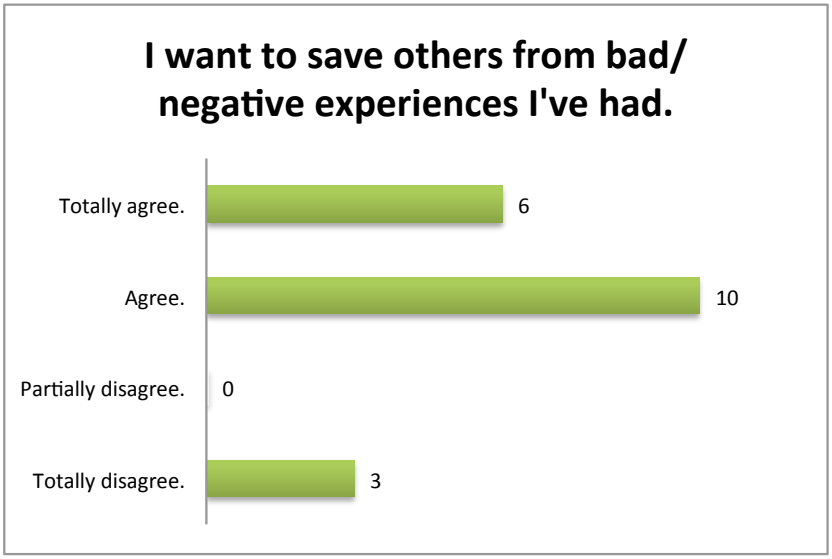


Figure 40. Willingness to share.
Source: self-elaborated.

There are mixed sentiments about current software done at organizations, respondents are not agreeing about it. This can be seen as an opportunity to use their willingness to share to improve KT process, and an opportunity to introduce KT as an SR success enabler.

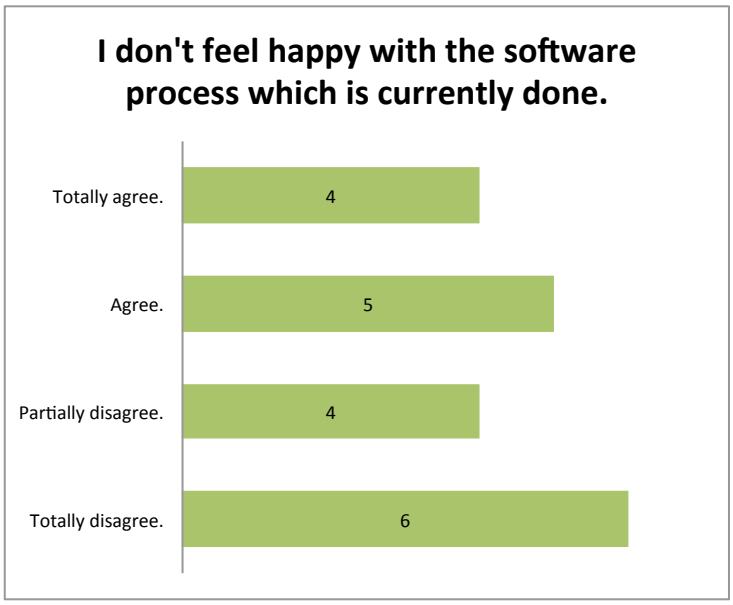


Figure 41. Software process pleasing.
Source: self-elaborated.

There was found a big agreement about share successful experiences.

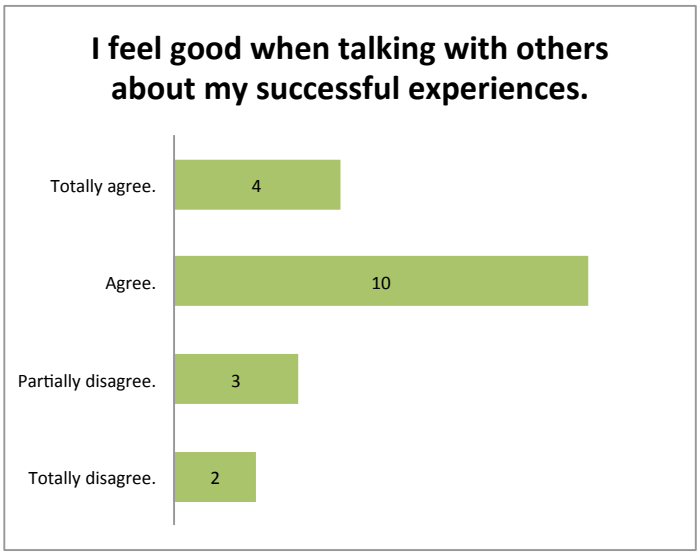


Figure 42. Motivation to share successful experiences.
Source: self-elaborated.

Something interesting is that not everyone has a strong desire to be rewarded for share their knowledge.

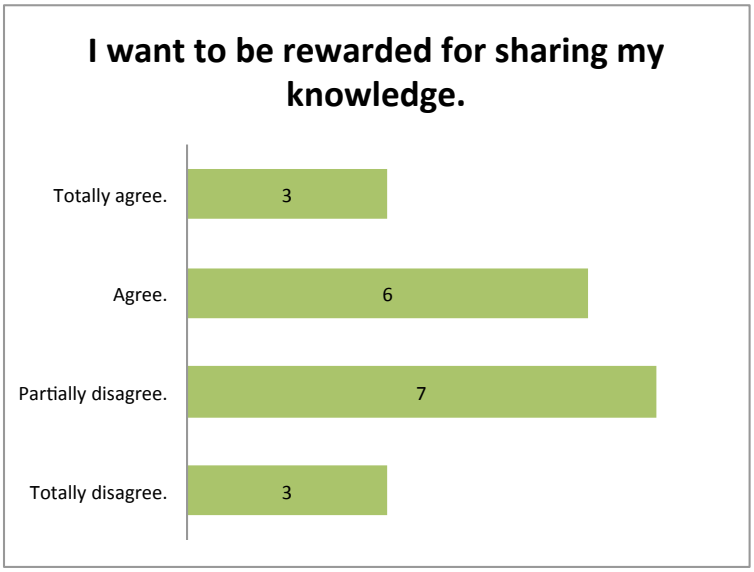


Figure 43. Intention to be rewarded due to KS.
Source: self-elaborated.

There is a strong agreement in responsiveness to share between people with similar interests.

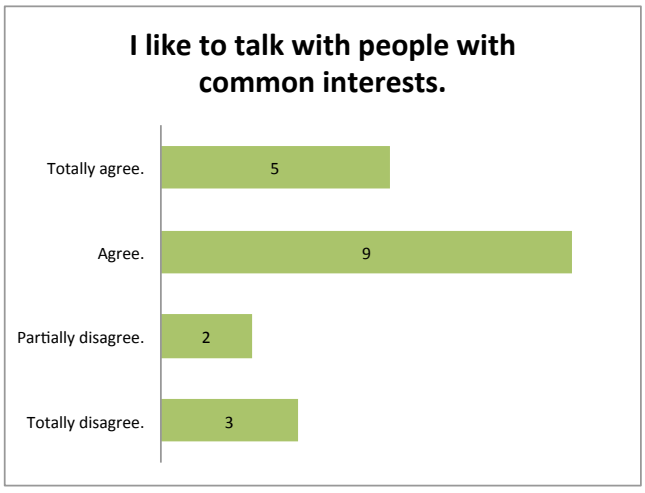


Figure 44. Empathy.
Source: self-elaborated.