UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Alenka Kolar

Zagotavljanje kakovosti projektov razvoja programske opreme pri zunanjih dobaviteljih

DOKTORSKA DISERTACIJA

Mentor: prof. dr. Franc Solina

Ljubljana, 2006

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Alenka Kolar

Quality Assurance of Outsourced Software Development Projects

Doctoral Dissertation in Computer and Information Science

Mentor: prof. dr. Franc Solina

Ljubljana, 2006

Thanks

Being grateful means you know special people you respect and can call them friends.

I owe special thanks to professor Solina and professor Jaakkola who carefully revisited my thesis and gave me direction to improve it.

To my mother and father because you believed that I can do it and supported me all the way.

To my friends because you inspired me with your stories.

LLMM

Contents

List of Figures

List of Tables

Zagotavljanje kakovosti projektov razvoja programske opreme pri zunanjih dobaviteljih

Povzetek

Doktorska disertacija je umeščena v okolje javnega gospodarskega podjetja Elektro-Slovenija d.o.o., hkrati pa zagotovi pregled podobnih gospodarskih javnih služb. Njihova skupna značilnost je, da razvoj programske opreme ni osnovna dejavnost podjetja ter da zagotavljanje podpogodbenikov omejuje zakon o javnih naročilih. V podjetjih je običajno ustanovljen oddelek ali sektor za informatiko, ki obvladuje informacijsko podporo za ostale zaposlence v podjetju. Informatiziranost poslovanja je zahteva vsakega sodobnega podjetja, le da se te naloge podjetja lotevajo na različne načine. V disertaciji je opisan način, ko podjetje sicer razpolaga z lastnimi zaposlenci, ki obvladajo izdelavo uporabniških rešitev, število zaposlencev pa ne zadošča v primeru večjih projektov razvoja programske opreme. V takih primerih sklene podjetje pogodbo o zagotavljanju storitev z zunanjim podjetjem (podjetji).

V veliki večini uporabniške rešitve nastanejo v projektnih skupinah, ki jih naročnik kupec imenuje za točno določen projekt z znanim ciljem in omejitvijo v času. Metode obvladovanja projektov (angl. *Project Management*) izhajajo iz dolgoletne tradicije gradbeništva in strojništva, zato niso v celoti primerne za področje razvoja programske opreme. Kakovost rezultata projekta razvoja programske opreme, uporabniške rešitve, je težko določljiva in je v disertaciji podana kot četrti dejavnik (poleg, časa, prvin poslovnega procesa (angl. *resources*) in stroškov), ki vplivajo na uspešnost projekta. Zagotavljanje kakovosti naj bi bilo vgrajeno že v sam proces izdelave uporabniške rešitve. Na tak način se izognemo končnim, ponavadi zelo dragim popravkom, ki jih zahteva uporabnik. Pri doseganju kakovosti nam pomagajo standardi in dobra praksa.

Prilagodljiva metoda razvoja programske opreme (angl. *Adaptive Software Development (ASD)*), je samo ena izmed metod agilnih pristopov, ki od leta 2000 predstavljajo velik del dobrih praks pri razvoju programske opreme. Značilnosti metode prilagodljivega razvoja programske opreme lahko pomagajo projektnim vodjem, da na opisan način organizirajo delo na projektih razvoja programske opreme in tako formalizirajo razvoj programske opreme. Metoda predpostavi, da so pri razvoju programske opreme najpomembnejši ljudje, disertacija pa skozi prikazan model ugotavlja, da gre pravzaprav za pomembnost razmerij med ljudmi.

Problematiki zagotavljanja zunanjih dobaviteljev razvijalcev programske opreme je posvečena poglobljena študija. Največja težava nastane, ker iz-

vajalcev ne moremo izbirati zgolj po subjektivnih kriterijih vodje projekta ali naročnika projekta znotraj podjetja kupca, ampak objektivno preko tehničnih kriterijev in še najbolj cene, ki jo dobavitelj ponudi. Dejavniki, ki so dejansko pomebni za ustvarjanje kakovostnih razmerij, pa nikakor niso pogojeni s tehnično specifikacijo ali ceno. V taki skupini, ki nastane z kombinacijo domačih programerjev in analitikov ter zunanjih programerjev postanejo izredno pomembna razmerja med člani skupine. Gre za težko določljiv in zelo oseben dejavnik vpliva na kakovost projektnega dela in s tem na kakovost uporabniške rešitve. Eden od problemov izbora članov projektne skupine je tudi, da so skupine v celoti neponovljive, tako kot projekti.

V disertaciji je uporabljena Metoda ugotavljanja kakovosti organizacije združb (MUKOZ), ki jo je leta 1988 razvila slovenska skupina strokovnjakov na Gospodarski zbornici Slovenije. Določili so pet osnovnih razmerij in kombinacijo parov teh razmerij, ki vplivajo na kakovost organizacije. Določena razmerja (razmerja kadrovske narave), ki jih predvidi omenjena metoda (MUKOZ), so pri delu na projektu razvoja programske opreme pomembnejša od drugih in lahko celo zagotovijo uspešnost projekta kljub morebitnim drugim (tehničnim) pomanjkljivostim. Pomembna razmerja temeljijo na osebnih in profesionalnih značilnostih članov projektne skupine, ter na lastnostih okolja, ki obdaja projektno skupino oziroma ga pogojuje uporabniška rešitev, ki jo skupina pripravlja.

Namen doktorske disertacije je združiti poznavanje podjetniške prakse v gospodarskih javnih podjetjih in teoretičnih znanj s področja obvladovanja projektov in zagotavljanja kakovosti za določitev dejavnikov, ki vplivajo na razmerja v projektni skupini in omogočajo nastanek projektne ekipe. Pri tem smo izhajali iz raziskave opravljene med dobavitelji programske opreme, ki v projekt kupca pripeljejo svoje zaposlence. Na vodji projekta je, da izkoristi ne samo znanje članov mešane projektne skupine, pač pa tudi njihove ostale lastnosti. Človeke osebne lastnosti omogočajo sinergijo učinkov in razvoj projektne ekipe. Disertacija opiše dvajset takih lastnosti, na katere mora biti projektni vodja pozoren, pozoren pa mora biti tudi naročnik projekta, ko določa projektnega vodjo. Lastnosti so utežene glede na njihov vpliv na razmerja, ki zagotavljajo prehod projektne skupine v projektno ekipo.

Idealna kombinacija človekovih lastnosti zagotovi razmerja, kjer prevladujeta zaupanje in samoiniciativa. V takih razmerah, kjer se organizacija kot sestava razmerij že nagiba k prijazni organizaciji, dobimo kakovostne uporabniške rešitve. Poleg projektnih formalnosti in uporabe dokumentacije ter standardov, predlagamo izrabo pozitivnih osebnih in profesionalnih lastnosti članov projektne skupine za doseganje kakovosti projektnega dela in rezultata projekta v projektnih skupinah, kjer sodelujejo člani projekta iz različnih okolij. Za uspeh dela projektnih skupin za razvoj programske opreme in njihovo preobrazbo v ekipe so odločilnega pomena razmerja med člani skupine.

Quality Assurance of Outsourced Software Development Projects

Abstract

This thesis is about software development in enterprises such as the public company Elektro-Slovenija d.o.o. whose core business is not software development. What is also common to these enterprises is that they are publicly owned and that they must follow public procurement laws. These enterprises have usually a department or a sector dedicated to information technology (IT). IT departments provide to other departments IT support, both computer hardware and software. User applications for the business information system are today the nerve system of every modern enterprise. Enterprises cope with this support in different ways. In this thesis, the following case is analyzed. The IT department has their own developers but not enough to cope if big software development projects are at stake.

Computer applications are in general a result of software development projects. A project is a process where a certain goal must be reached in a certain amount of time using limited resources. Project management has its roots in civil and mechanical engineering. The project management methodology is therefore not entirely suitable for software engineering projects. The results of software engineering projects are computer applications which are far more intangible that the results of other engineering projects. The quality of the end product is in the case of software engineering projects hard to predict and is often regarded as the fourth crucial factor, besides time, resources and cost, influencing the success of a project. Quality assurance should be predetermined and built into the software development process itself. In this way, user demanded late changes of software can be prevented. Different standards and best practice approaches in software development can help in achieving quality.

Adaptive Software Development (ASD) method is only one of agile approaches to software development which are since 2000 becoming very popular. The ASD method can help project managers to organize the work and assign the prescribed roles in the project team. Since changes are a constant in today's business, ASD anticipates the importance of the people on the project, because they are the one that can adapt to ever present changes. This thesis is built upon this presumption and tries to prove the following hypothesis that relationships between project team members are capable of adaptation to change.

Outsourcing part of the software development to sub-contractors is studied in detail. The biggest problem arises because the subcontractor cannot be selected based on the purely subjective choice of the project manager, but through some objective technical and economic criteria, price being one

of the most important. In a project team that is composed of in-house programmers and analysts, and programmers that come from a subcontractor, the relationships established between team members gain on their importance. This thesis tries to prove that relationships between people influence the quality of the finished product but this factor is hard to predict and is of personal nature.

In this thesis the methodology of assessing the quality of organization in an enterprise (MUKOZ) is used. MUKOZ was developed by Slovenian experts in the Slovenian chamber of commerce in 1988. This methodology established five basic relationships and combinations of pairs of them that influence the quality of organizations. Some of these relationships (of personnel nature) are in software development more important than others and can even surmount other quality factors of more technical nature. Relationships of personnel nature are founded on personal and professional characteristics of project team members on the environment in which the project is carried out, and on the software applications itself.

The purpose of this thesis is to combine the knowledge of everyday business practice in a Slovenian publicly owned enterprise with theory of project management and quality assurance. The goal of this thesis is to single out the factors that are the most influential for the personnel relationships in the project team during software development. A survey among sub-contracting software developers was carried out to determine these factors. A project manager must exploit not only the technical knowledge of his team members but also their other characteristics. Personal characteristics of team members enable the synergy and development of a project team. In the thesis twenty such characteristics are identified that team managers must take into consideration. The characteristics are weighted according to their influence on building the project team.

An ideal combination of characteristics assures relationships where trust and self-initiative prevail. In such circumstances the organization as a structure of relationships turns toward a more friendly organization. A friendly organization produces solutions of best quality. Beside proper use of documentation and standards we propose for quality results also the use of positive personal and professional characteristics of project team members. This is even more important in heterogeneous groups where team members come from different environments. For the success of a software development project, relationships in the project team play a vital role.

Chapter 1

Introduction

Quality, Project Management and Outsourcing, are three issues highly critical for the kind of software application development which is the subject of this thesis. Although they present three distinct fields, they often overlap in practice. Software applications used in an enterprise are becoming more sophisticated and need fewer employees to input data. They support several processes with one application from start to finish and more managerial decisions are made on the basis of information gathered in an IT system. In short, IT systems are becoming the backbone of the business process. Similarly, the same backbone integrity that we have come to expect from the human contribution, is also expected from software applications.

Developing software is different from building a bridge in civil engineering or making a gear wheel in mechanical engineering. Products of engineering can be "easily" measured, repeated and there is usually only one way to do it best. None of these three characteristics are true to software development. It might look like exaggeration, but the results presented in Figure 1.1 are sadly often true.

"We don't notice all that is in front of us, and we don't have adequate names for what we do notice. But it gets worse: When we go to communicate, we don't even know exactly what it is we mean to communicate [15]." This statement often applies to software development projects when outsourcing is used. The users, demand quality software applications, yet they don't know how to transform this requirement into a form that sub-contractors would correctly interpret.

This thesis is focused on problems that arise when software is developed in project groups consisting of in-house developers and outsourced members (team members employed at sub-contractors) and where the final user is well known and is present all the time during the development. Majority of enterprises in Slovenia develops their own information systems or at least customize standardized systems like SAP and Oracle e-Business Suit. The development teams during that process are often composed of people from

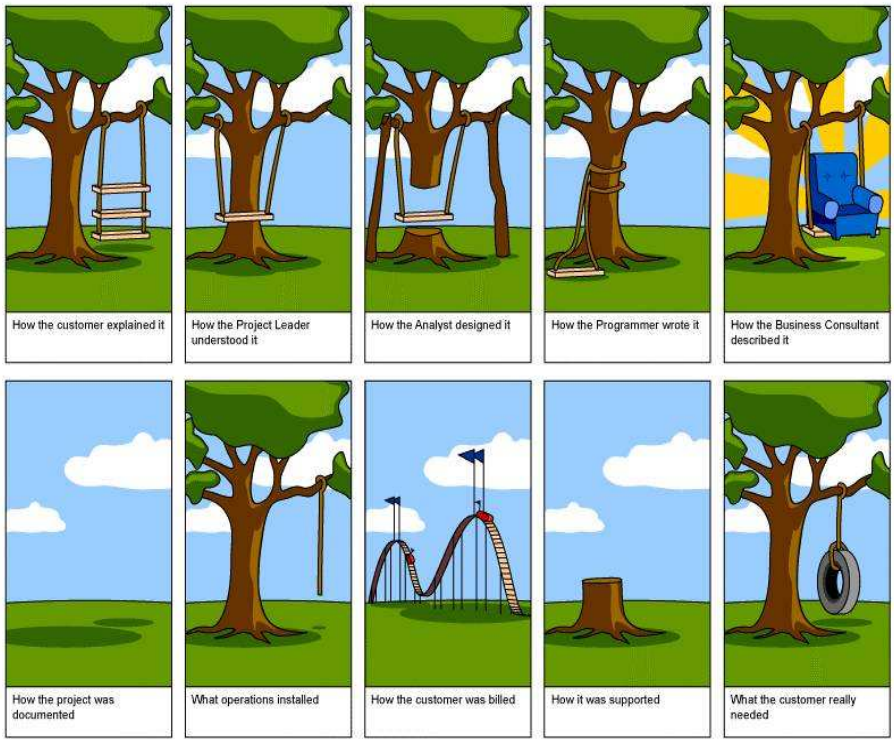

Figure 1.1: How software development is viewed

different origins (in-house and outsourced from subcontractor who performs the outsourced activities). In particular, this thesis tries to define methods that enterprises need to ensure quality of such projects.

1.1 Definition of terms

In this section the most important terms related to the topic of this thesis are explained. Quality and project management are topics of chapters 2 and 3, respectively.

Enterprise or Company; A group of people joined together and sharing work to reach a final common goal(s); A form of connecting people that enables realization of goals in the society that could not be accomplished by individuals. The word Firm is not used in this sense, but only to identify the name of the company (e.g. ELES).

Organization and relationships; Organization in this work is meant strictly as a structure of relationships between people (that become, through this connection, members of the same company) this ensures the existence of special characteristics within this group of people and the suitableness of the

goals that have been assigned to this group. Also, a relatively autonomous group of people and means that through a defined set of activities, aims to reach a set of goals which satisfies a need.

Business informatics (*business information system*); Applications supporting processes like accounting, finance, human resource management, ordering, contracts, financial tracking of projects, billing, energy accounting and cash desk.

Outsourcing (or **contracting out**); A way of conducting business by relocating some of the activities to be done, outside the enterprise. In the case of this thesis, programming the code for business applications by subcontractors who perform the outsourced activities. One can also outsource analytical functions, but then the implementation of the new application can be prolonged because of the ever present danger of misunderstandings that exist between users and analysts/programmers. Outsourcing is complex and strategic, and executives need a breadth of skills and a depth of knowledge to do it successfully [21]. Defined also as the delegation of non-core operations or jobs from internal production within a business to an external entity that specializes in that operation. Outsourcing always involves a considerable degree of two-way information exchange, co-ordination, and trust. [1]

Public procurement; Purchasing material and services by publicly owned enterprises. Every purchase over a prescribed amount must be announced in public and offers must be gathered. Only the lowest bid by price gets the business if technical conditions are met.

Best practice; A technique or methodology that, through experience and research, has proven to be reliable in achieving a desired result. In software development, a best practice is a well defined method that contributes to a successful step in product development. Three main barriers to adoption of best practice are a lack of knowledge about current best practices, a lack of motivation to make changes involved in their adoption and a lack of knowledge and skills required to do so.

1.2 The problem domain of this thesis

The main subject of this thesis can be described with three terms: ***quality, project management*** *and* ***outsourcing***. To every one of these terms many books and articles have been devoted. This work tackles all three mentioned subjects, but only as they apply to the field of software development in enterprises where software applications support everyday business. The author of this thesis has studied this problem domain extensively as the head of department for developing and maintaining business informatics applications in Elektro-Slovenija d.o.o. (ELES) [27, 30, 31, 32].

[1] Definition from http://en.wikipedia.org

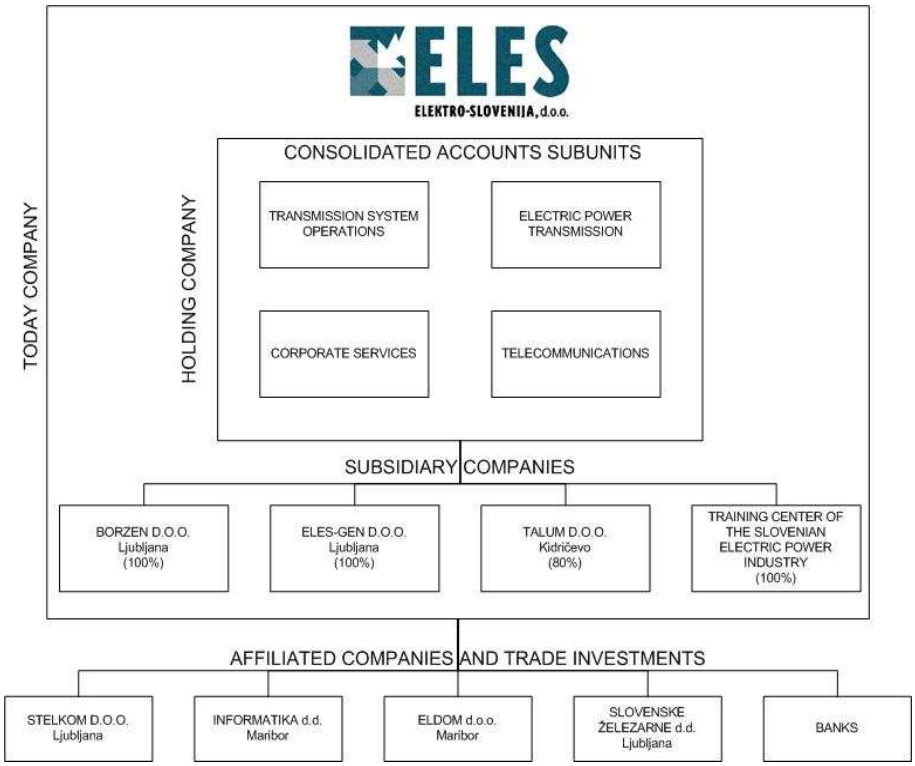

Figure 1.2: Organization in ELES

ELES has 500 employees, and its core business is transmitting electricity on high voltage lines. ELES-Slovenija is viewed as a relatively large company in Slovenia. The organization chart of the company is presented in Figure 1.2.

ELES is the only power utility company in Slovenia. They build, maintain and control 400 kV, 220 kV and 110 kV power network. The Slovene transmission grid comprises 1756 km of overhead lines on 1110 kV, 328 km on 220 kV, and 510 km on 400 kV level. In the high voltage grid there are 8 transformer sub-stations with 17 transformers 400/x kV (2700 MVA) and 220/115 kV (1500 MVA) [33]. Besides that, ELES maintains and controls telecommunication lines for their own needs to control the power flow. They also sell some excess capacity from their telecommunication network on the open market. Business informatics runs Oracle 10g with WEB access and supports business functions needed for the company. Process control systems supply national transmission system operator (TSO) with large amounts of data from the power utility network. The fact is that they need all this data from both the power utility and the telecommunication network to satisfy Slovenian needs for electric power. To connect different parts of information system they use their own IP/VPN/MPLS network.

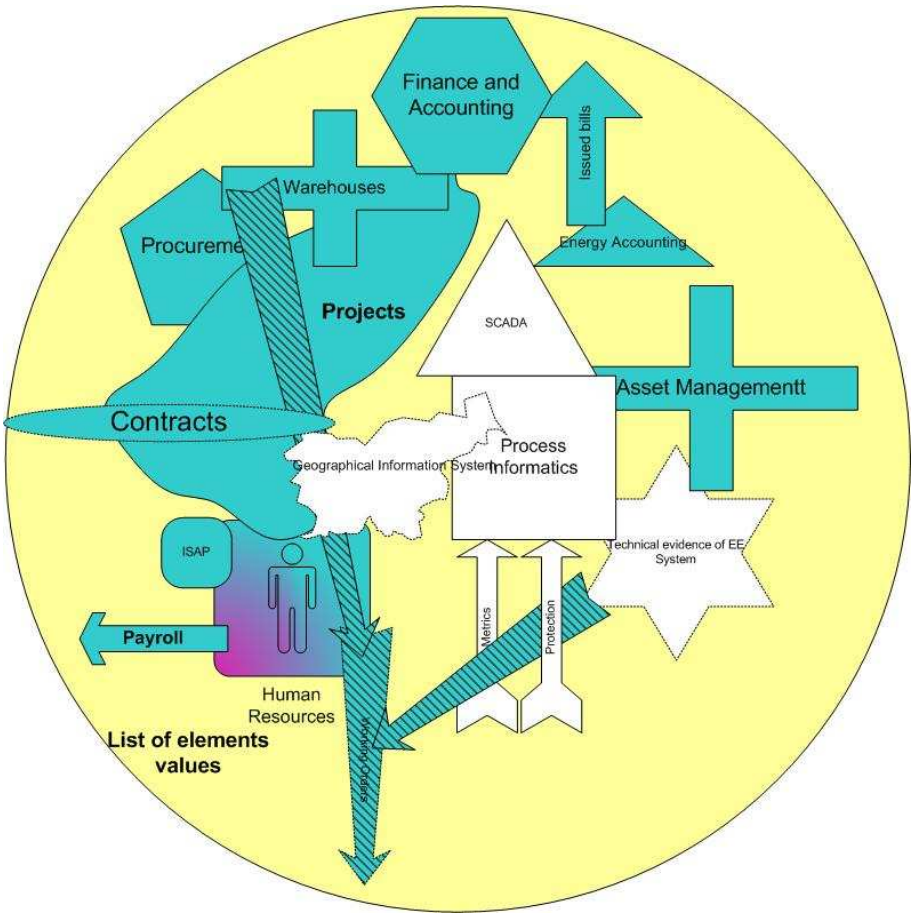

Figure 1.3: Main business application and their relationships in ELES - possible EAI

The Business informatics department has 18 employees including 4 programmers/developers and 3 field analysts which is less then 4% of the total staff. The enterprise is using an Oracle database and the Oracle Developer Suite to code applications in business informatics. The department manages 40 applications composed of 1200 forms and 900 reports. The connections between applications are shown in Figure 1.3. These applications are:

1. Warehouse
2. Material accounting
3. General Ledger Posting
4. Documentation of projects

5. Planning and analyzing
6. Cash desk
7. Cash desk - foreign currencies
8. General Ledger
9. Compensations
10. Cost Account Management
11. Assets
12. Assets - guardians
13. Small assets
14. Interests
15. Account Payable
16. Current Account book
17. Current Account book - foreign currencies
18. Buyers
19. Suppliers
20. Human Resource Management (HRM)
21. Civil Defense
22. Orders
23. Accounts Issued
24. Value Added Tax
25. Look-up tables
26. Customs
27. Salary
28. Cash flow
29. Construction In Progress (CIP)
30. Projects
31. Business Cars Management

32. Energy Accounting
33. Contracts
34. Reminding
35. Traveler's Orders
36. Apartments
37. Cost Carriers
38. Library
39. Supervision
40. Technical Evidence (Spare Parts and Asset Components)

Enterprise application integration (EAI) is the next logical step for the enterprise in question [27, 33]. Applications that are part of business information system are represented with blue color, white are applications in process informatics.

The Oracle database supports 200 concurrent users. To be able to cope with everyday changes of legislation and to cope with demands of other departments in ELES (accounting department, financial department, procurement, project managers, secretaries, etc.) they collaborate with up to 10 sub-contractors, different independent software companies, to whom they outsource part of the development.

In Figure 1.4 the process of changing and developing applications is represented and entry points where outsourcing comes into the process are specifically marked. Process charts are usually best viewed as work-flow diagrams, like this one describing *who* receives *what* from *whom*.

In the last few years outsourcing was the "buzz" word in many articles, at many conferences and even in books [69, 26, 21, 52]. Enterprises like ELES where the development of software applications is the result of a complex interaction of in-house personnel and outside experts by way of outsourcing are not rare in the business environment. The constraints in the case of ELES that make the software development even more difficult are:

1. Public procurement law [94].
2. Software development personnel are not considered strategic personnel.
3. Functional services in the enterprise are strong and hold the domain specific knowledge within their "fences".
4. Sub-contractors press hard to do business with the publicly owned enterprises that have a good reputation for being "a reliable payer".

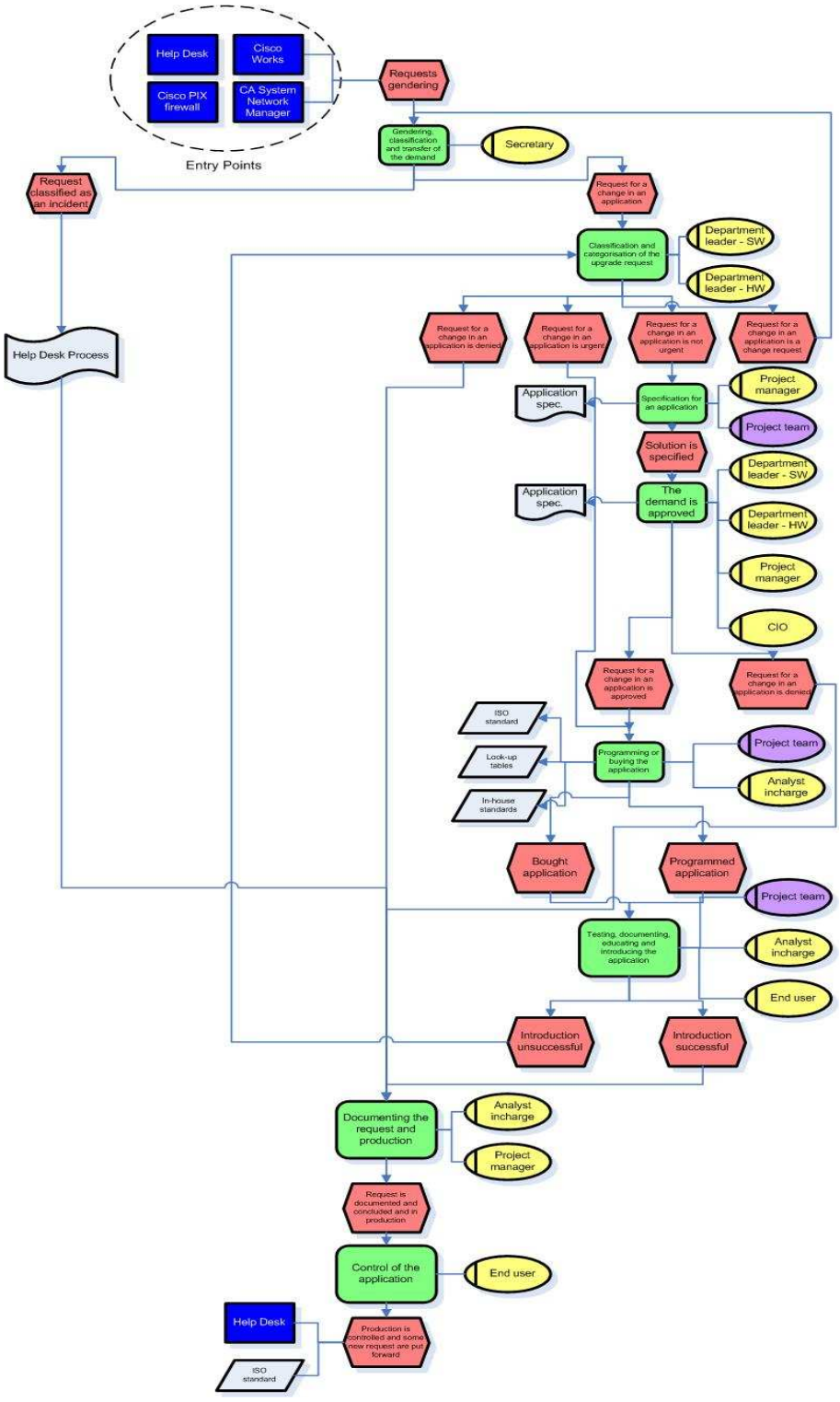

Figure 1.4: Process of Change Management in IT in ELES. On the top are shown the entry points for outsourcing activities

5. Policy and strategy of software development in the enterprise is either not well defined or is constantly changing (political changes in government).
6. Employing new personnel is tightly controlled or difficult.
7. Personnel lacking in motivation cannot be changed or removed from their posts.

1.2.1 Core business and what can or should be outsourced

Small businesses such as ELES whose core business is not development of software usually outsource at least part of that process. However, analysis of the processes should stay in-house or the deployment of that software can be extremely difficult. Keeping in mind that we are not talking about adapting the processes and the organization to the applications, but instead adopting (as far as it is reasonable) applications to the processes and organization found in-house. Enterprises can outsource complete functions such as accounting and then, of course, there is no need to informatize such a process. Core processes cannot be outsourced and in today's world they just yearn for informatization.

The scope of informatization depends, among other things, on the IT team, their number and their knowledge. Some processes that are going to be informatized, are very unique to the company, others are more common. Sometimes universal processes are easier to be outsourced for programming, because there is enough knowledge around and internal resources can be used for other more bespoke processes. The other case is that even some very special processes are sometimes standardized globally, and some organizations outside the enterprise under review, prescribe certain solutions (case: CIM (Common Information Model) for EMS Energy Management System) represented in Figure 1.5. Common Information Model describes and foresees the main part of data model that describes the energy system in a certain area (country). Prescribed data model is basically the same for any energy transmitting enterprise.

It therefore depends on the given case and actual environment (budget, time, resources), to decide what will be designed and programmed inside or outside the company. Here are some of the considerations [42]:

1. Reuse
2. Staffing flexibility
3. Experience
4. Better requirement specification
5. Reduce feature creep

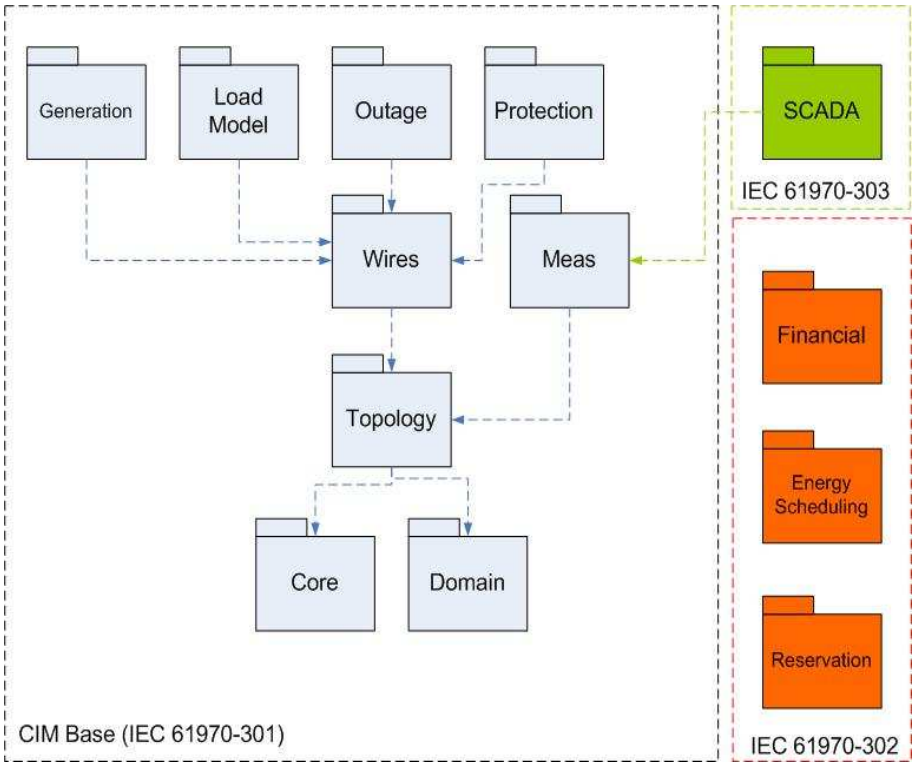

Figure 1.5: Common information model for an energy utility company

6. Develop a management plan including risk management
7. Learn about contract management
8. Make communication with the vendor a priority
9. Count on using some of your own technical resources
10. Be wary of unstable requirements
11. Retain enough control to pull the work back in-house if needed
12. Especially consider outsourcing legacy systems (reengineering)
13. Avoid double standards for outsourced work

1.3 Goals of the research

Every enterprise that has to outsource at least some of its development effort in the IT field has to cope with the same questions. How to satisfy the needs of our customers? Can we accurately and on time, transfer their wishes into

the program code? How to get the best from the outsourced services with the minimum internal effort? How to present the demands of the customer to the developers? Is there a "best practice" form of communicating the customer's needs to the designers/developers?

A unified model of how to work with people on both sides of the process of coding the application is necessary. If one is used to the notion that it is the service providers that ask their customers if they are satisfied with their services or product, I shall instead ask from the buyer's perspective, if the service providers are satisfied with the company for whom they work. My hypothesis is that the relationships that are formed between the staff on the buyers' side and the developers in the subcontractor company who do the outsourced work, significantly influence the quality of the project work and the quality of the project product. Typical projects in the IT department are small and cost approximately 125 000 EUR per year and therefore do not require a very formal organization. Nevertheless, these kinds of projects bring a return on investment (ROI) that is in the long run far greater that the money invested.

The goal of the thesis is therefore to formulate an approach that will combine established quality assurance and project management techniques in a framework that will enable the cooperation of in-house and outside developers without excessive formalization.

The project teams are composed of members that come from at least two enterprises. The hierarchical line of "command" is not always clear, although one can suppose that the buyer's side should be stronger in making decisions about the goal of the software development project. The overall methodology should be viewed as a "best practice" approach.

1.4 Purpose of the thesis

Combining knowledge from the fields of informatics, quality assurance, project management, organizational relationships and experiences on providing quality products in the field of business informatics, one should be able to achieve lower costs, higher quality and the optimal use of resources. Is there, in general, the best way to achieve that goal? What are the circumstances influencing this process and can they be controlled?

The purpose of the thesis is to find out how to get the "quality of service" when coding the application at a sub-contracted company. How to actually direct actions that lead to the desired quality. To be able to do that, we have to know what are the possible actions that need to be taken, at what moment and by whom. Furthermore, since services are consumed and not possessed, the performance of those providing the service offers some clues as to the quality of service [9].

Although the adage "If at first you don't succeed, try again" is a great

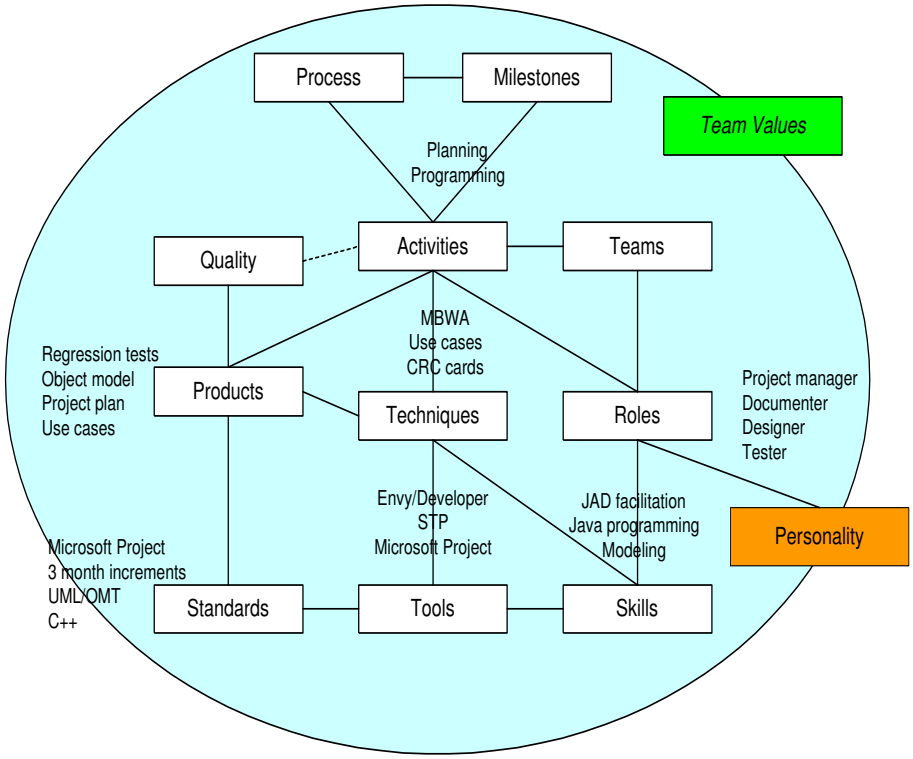

Figure 1.6: Elements of a methodology

motivator, it is not an effective approach for achieving goals in outsourcing. Negative situations can spin out of control. Absorbed in the details, the parties find themselves facing an ever-growing snowball of dissatisfaction. Failure to achieve the desired end results can lead to the cost-prohibitive process of terminating an unsuccessful relationship and finding another service provider [21].

1.5 The methodological framework

Methodology combines everything that we regularly do to achieve the result we want. No single methodology definition can possibly suit all projects. The result of a methodology used on a project can be either a product or a service. I shall look for a methodology that combines people in the teams with quality and yields a quality product. If methodology is represented with Figure 1.6, there are several elements we shall look at in detail. Those elements that will be in the center of the study are: Teams, Roles, Quality, Standards, Personality and Team values. The last two elements are in Figure 1.6 on the border of this particular methodology. They surmount any particular

methodology.

Methodology combines philosophy, principles, ideas and thoughts of people that form a project team or an enterprise. Authors in [66, 67] are looking for a software development methodology evaluation method by exploring two dimensions: technical suitability and social suitability of the methodology. Methodology is formed by the people for the people or the group of people, in this case the team that develops software. Some parts of methodology can be formalized, other procedures are undocumented and unformalized. Knowledge of the team is one element that is very difficult to put on paper, the other even more tacit building element is the **relationship formed between team members**. Out of informal elements in time comes formalization that is again the basis for new learning. This cycle of dependencies between formal and informal elements is shown in Figure 1.7.

Large amount of formality in work starts to slow down the development if we want to keep papers and procedures up to date. They have to be updated regularly and that takes employees away from the main work which is software development. If procedures are strictly informal they become hard to repeat and oversee. Maintainability of the system becomes problematic. When the system fails, the personnel may spend most of the repair time in diagnosing the problem, for it may take considerable effort to understand the nature of the failure well enough to be able to locate the cause [36]. Developing software is therefore caught between fast and reliable and between improvisation and standardization. This work shall look into formal procedures and find the informal solutions.

The following sections look in to new trends and approaches. The red line of every described approach in this thesis is that the project team members come from inside the enterprise and from a subcontractor who does the outsourced work.

1.5.1 Agile Software development

Today change is the only constant. This is the fact software development must adapt to. In year 2000 a grip of developers[2] wrote down a manifesto that should bring agility to best practices in software development. Cockburn as one of them in his work [15] sets four postulates of developing software in an agile way:

1. Individuals and interactions over processes and tools

2. Working software over comprehensive documentation

[2] Alistair Cockburn, Kent Beck, Mike Beedle, Arie van Bennekum, James A. Highsmith, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland and Dave Thomas.

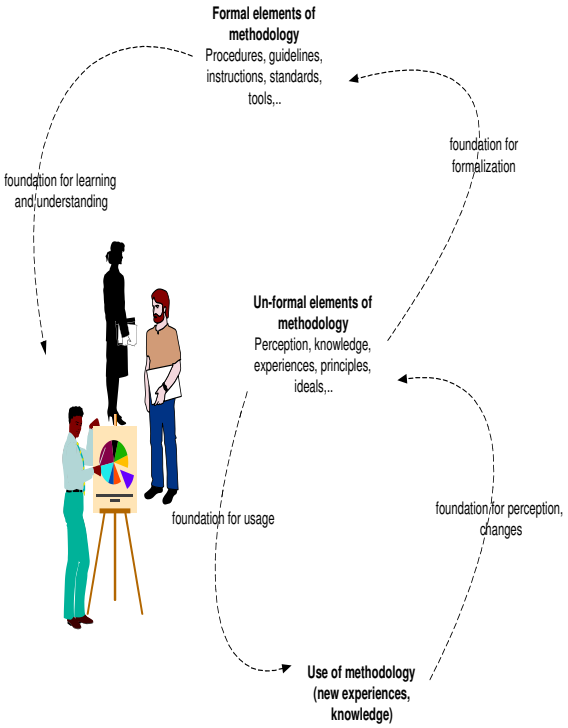

Figure 1.7: From un-formal to formal elements of methodology [4]

3. Customer collaboration over contract negotiation

4. Responding to change over following a plan

Out of these four named rules, organizations like ELES can benefit if the requirements of some world renowned quality standards can be applied or at least adapted. Authors in [4] describe agile software development methodology for a software development company. Agile software development is a step away from standardized procedures. Because of the dynamics and ever faster competitors in this market, speed and adaptability go hand in hand with improvisation. Quality and reliability must nevertheless be assured.

Results are better if communication between team members is good rather than if every procedure is put on paper. The most efficient and effective method of conveying information to and within a development team is face to face conversation. Communication will be described in more detail in Section 5.3. The final user needs primarily, an application that works as it is supposed to do, the associated documentation is less important to the user. Documentation is still important, but it is usually more useful for the developers to have everything documented and repeatable. For the user

the documentation's added value shows up only if something goes wrong. A possible solution for documentation is described in Section 2.8.1. Documents can be very useful, as we will see, but they should be used along with the words "just enough" and "barely sufficient" [15]. This work not only presents two sided collaboration between final user/customer and the software developing team, but the latter are further divided down into in-house developers and the outsourced ones. Chapter 4 is devoted to this additional relationship. Project plans that define activities in time are useful, but must allow changes. Developing software is not as predefined as building a bridge or producing a gear. Differences will be highlighted in Section 3.6. Time must not be our enemy but the means to reach the goal.

Abramsson, Salo, Ronkainen and Warsta [1] made a review and analysis of the currently existing agile software development methods. They are as follows: Extreme programming (XP), Scrum, Crystal family and methodologies, Feature Driven Development, The rational unified process, Dynamic system development method, Adaptive software development and Open source software development. Developing software in a company such as ELES comes close to the process of adaptive software development, so this method will be looked at more closely later in this work.

1.5.2 Best practices

The Gartner Group's research [91] gives us the list of factors forming the best practice in managing information technology:

- Standardized software and hardware
- Net shared printers
- Help-Desk and Help Desk call analysis
- Users must be familiar with the IT value of their area of work
- Regular communication between business units
- Educated users
- Limited network capability
- Limited use of Internet

This thesis will look only into some of the above listed factors or even try to deny some. One cannot just take the ISO standard and the quality system of a successful enterprise and replicate it in another enterprise to make it successful. We can try to adopt some parts of the methodology that an enterprise uses and that we think makes them successful. ELES has its own personnel, team members, built-in thinking, knowledge and relationships.

Before starting a project, one can reconsider some of the best practices in the field of outsourcing, rethink and adopt some of the other practices used in software development.

1.6 Contributions of the thesis

In Section 1.2 Elektro-Slovenija d.o.o. (ELES) is being described as a relatively large enterprise in Slovenia. Given the turnover and the value of the enterprise assets, ELES is, by Slovenian standards considered a large enterprise. However, the enterprise employs only 500 people and by this measure alone it cannot be considered big. Given world standards, enterprises like ELES can be considered small or in some cases medium sized. Business informatics department in ELES forms less then 4% of all staff. The company does not have any affiliates outside the country and their projects are not multinational. The number of their final products or services is less then 5. Even measured by data input into the business informatics database, the enterprise still remains small.

Quality depends on many known and unknown factors. Some can be predicted, others can not be changed. The role of a project manager is to get the best people for the job and to combine them in the best possible way with as little friction as possible. How can the behavior of the team be predicted? What are the circumstances influencing this process? The result must be quality software delivered on time. Some things can be done in advance (using standards where applicable, testing software, getting the personnel with the right knowledge, etc.), the rest must be directed using appropriate methods and approaches.

My hypothesis that I will try to prove in this dissertation are:

1. *Relationships formed in a project team before and during development of an information system or application are important factors influencing the quality of the final product, but hard to plan and predict in advance.*

2. *The combination of ideal characteristics of project team members (personal and professional) yields better products.*

3. *Adaptive software development method is highly suitable for the environment which is the object of study in this dissertation.*

1.7 Structure of the thesis

In the Introduction the motivation for the dissertation was presented, the goals for research are set, and the hypothesis of this thesis is outlined.

In Chapter 2, "quality" as a term and as a movement is looked into. Standards that try to prescribe quality of processes and products are named, experiences from elsewhere in the world and some synonyms are also described. Chapter 3 is on project management. Differences between various projects types depending on their goals and different approaches to project management methodology are also presented. Chapter 4 is on outsourcing in general and on outsourcing software development in particular. Chapter 5 presents the survey done on outsourcing partners (sub-contractors) in the enterprise under review. The analysis of the survey serves to propose the right methodological approach for software development outsourcing for this type of enterprises. Chapter 6 concludes the thesis and summarizes the main contributions.

Chapter 2

Quality

Quality is a subjective term relating to expectations of a person, towards another person, product or service. It is a distinctive characteristics or properties of a person, object, process, product or service[1]. Quality is the other term for the ability to satisfy stated or implied needs. If a service or a product has the quality characteristics it is acceptable or suitable for a purpose given in advance.

2.1 Intangibility of service versus tangible product

The research of Bowen and Ford [10] indicates that there are a number of important and defendable differences between managing a manufacturing firm and a service. It is reasonable to expect that there are differences between managing an organization that produces something that can be seen, touched and held, and managing something an organization that produces something that is perceived, sensed and experienced. In case of service the assessment of the quality and value of the organization's output lies entirely in the mind of the customer.

Producing software applications can not be described as manufacturing goods, neither it is described purely as a service. The activity lacks something of each or on the other hand has something of both worlds.

Software production as manufacturing goods:

1. Final product (software) does not disappear as service does once it is completed.

2. Product (software) "manufacturing" is repeatable.

[1] The term "Quality of service" or QoS is in telecommunication community reserved for the probability of the telecommunication network meeting a given traffic contract or a probability of a packet succeeding in passing between two points in the network. QoS will in the following text not be used as stated in this comment.

Software production as a service:

1. Product as described in this thesis can not be wrapped up, sold from the shelf and used in every environment.

2. Results of the service are not repeatable.

2.2 Quality of software as an increasingly important factor in a software development project

Quality is the main trend in economy and even in society for the last 30 years. Product control as a first stage in quality assurance was first introduced during the industrial revolution. It is too expensive to allow failures and to leave their occurrence purely to chance and eliminate them in a later (final) production stage. Nowadays, we intend to prevent failures and to build in quality when our product is still on the drawing board. It is reasonable to know even then, what the quality of the product will be. But can we satisfy our customer's needs fully without making our company bankrupt?

The findings reported in [55] are based on a survey of project managers. The authors identified six components of software development risk and suggested ways of addressing them. The six risk components are "scheduling", "system functionality", "subcontracting", "requirements management", "resource usage and performance", and "personnel management". All of the risks came into consideration even more obviously when an in-house IT department needs to use outsourcing to develop an application. On such a project, the personnel that is used for development is coming from different environments and their project roles are often not clearly defined.

The foundation of the project management point of view is the so called project triangle (time, resources and costs) [49]. If we enlarge or reduce one of the factors, the equilateral triangle and with it our primary plan collapses. Quality represents the factor that doesn't allow disfigurement of the triangle or it can be understood as the fourth factor that makes a triangle a square. These factors are presented graphically in Figure 2.1.

For software development project Brook's law [12] is also valid and it reads as follows: *"Putting additional people on the project that is late, additionally puts the project behind time"*. New team members are not on the spot qualified for the job, they need training and with that they burden the rest of the team. Additionally to that in a larger team more time is necessary for communication and less time is left for work.

Today, one cannot sell a low quality product to a demanding customer. They are prepared to pay a reasonable price if they can later avoid possible high costs for maintenance and repairs [59].

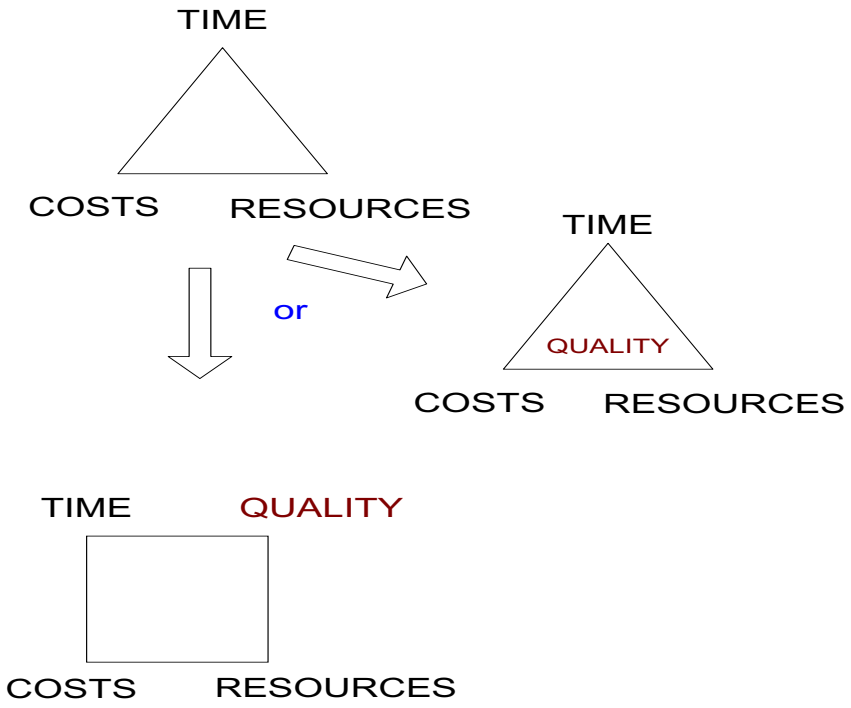

Figure 2.1: Factors influencing projects

2.2.1 What is quality?

Quality (like beauty) is in the eyes of the beholder. It is hard to define all the factors influencing the quality of software. It is even more difficult to measure it. Customer will soon tell you if what you have delivered is or is not the quality that he expected. It is almost impossible to get this expectation in a written form before the job on software development starts.

Many objects in a design come from the analysis model. Strict modeling of the real world leads to a system that reflects today's realities but not necessarily tomorrow's [20]. But even today's realities are often not described as "it is" but more as it "should be". The model reflects the wishes of people involved in a process. I have found the same situation when assessing, as an internal quality assessor, our ISO 9000/2000 quality system.

Characteristics of the business process could be presented with use case diagrams [40]. In this case, every step in the process would correspond to

one use case and to the employee that executes that step. Dependencies in the use case diagram would define the expected flow of business events. The flow of events could then be formalized with rules for transitions. Because of the business flow and complexity of transition rules, use case diagrams are not a convenient technique to represent business processes [71]. In this case we will not support the process for the benefits it could bring to the business, instead we support the process in the way that every current employee gets to keep his/hers job.

Employees involved in a process will say that software is of a high quality if their job has not changed and they have less to do. Whether or not their predecessor or successor has been buried beneath pile of paperwork is not their business. Users only care abut their part of the process. They expect useful written or on-line instructions and a quick response when something goes wrong. It can come so far that they even presume computers or programs will think for them.

Analysts have the thankless job of conveying the often unrealistic wishes of the end users into a language that programmers can understand.

2.2.2 Quality of software applications

It is hard to define all factors influencing the quality of software. It is even harder to measure it. Every customer will quickly tell you if, what you have delivered to him/her, is or is not of the quality that he/she had expected. It is almost impossible to get this expectation stated in a form before the software development actually starts. Software quality can be conceptually decomposed to quality factors and quality attributes. The three-level Quality Factor Hierarchy [41] in Figure 2.2 illustrates the structuring of quality attributes, which are in turn related to various measures or metrics. For analytical clarity, the attributes should be non-overlapping with respect to the quality factors, but this is in practice almost impossible to achieve for quality objectives stated in broad terms. Therefore, some attributes will be joined for a certain quality factor in the hierarchy (see Figure 2.2 [56]).

The use of expert group decision making to establish the quality of software rather than just the empirical evidence alone is proposed in [56]. In the case of uncertainty, the result is expressed by using the Triangular distribution. The use of statistics can give some answers, but the problem remains in the questions. Can conditions for reproducibility of measurement be controlled and repeated, and can judgment not be subjected to human influence? We have to ask ourselves - what are we assessing? What is important for the programmer may not be important for the user and vice versa.

Many objects in a design come from the analysis model. Strict modeling of the real world leads to a system that can reflect today's realities but not necessarily tomorrow's [20].

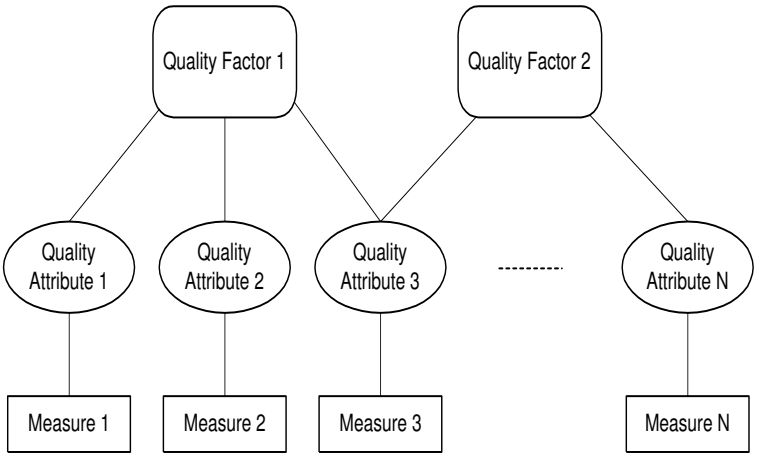

Figure 2.2: The quality factor hierarchy, where a quality factor is conceptually decomposed into Quality attributes and metrics [56]

2.2.3 Standardizing quality

Standards that are available in the field of software engineering mainly deal with how to standardize documentation that is produced along the process of making software.

Traceability is the most important characteristic of high quality software. This characteristic is the one that can be ensured with the help of documentation. With traceability we gain maintainability but often lose valuable time to develop software.

2.2.4 Service level agreement - SLA

Service level agreement or SLA is a written agreement between IT Service Provider (can be in-house or outsourced or combination of both) and the IT Customer(s), defining the key service targets and responsibilities of both parties. A true partnership should be develop between the IT provider and the Customer, so that mutually beneficial agreement is reached, otherwise the SLA could quickly fall into disrepute and a culture of blame prevent any true service quality improvements from taking place [87].

To be able to write down a Service Level Agreement it is necessary to look for the foundation deep in a business process run by an enterprise. The more freedom the customer has to define its own service, the more detailed is the charging structure. That means that before we can put SLA on paper IT must know its own costs, depreciation, return on investment (ROI), number of personnel needed to perform certain service, other indirect and direct costs.

2.3 Quality standards

Ever larger demands for fast but quality development of new custom made software are designated for project management type work. Projects are in their very definition something that occurs only once and is not completely repeatable. How can we then standardize something that is in advance destined to be different from what we know from our past experiences [34]? Existing standards in the field of software engineering [22, 65] deal with three different views on software. These are vocabulary standards, product standards and process standards. The last one is mainly connected with well known ISO 9000 series of standards and does not completely cover project management. Sadly, standards like ISO/IEC 12207:1995, ISO/IEC 15910:1999, ISO/IEC 9294:1990, ISO/IEC 15846:1998, ISO/IEC 6592:2000 and many others are not used in ELES or in the subcontractor companies who do for ELES the outsourced work. Because ITIL [87] (*Information Technology Infrastructure Library*) is becoming more and more interesting in the word of software development maybe ISO 20000:2005 will be more penetrating in Slovenian enterprises. It is therefore impossible to discuss "quality of software" because there is nothing we can compare software with. If we "a priori" do not comply with a standard, we therefore have no "measuring instrument" that would tell us how good our process and our products actually are. The consequences of a chaotic development can be felt every day, not only by the users of software products but also by developers that use a large amount of their time to correct past mistakes. We often don't have the time to do it right the first time, but "we have" time for endless corrective actions. If standards are not applicable as they are written, we could adapt them if they are not very strict in how things should be done. We could upgrade them with the knowledge gleaned from other projects and the key characteristics of project management.

ISO 9000/2000 and TickIT

In the early 1980s, the International Standard Organization (ISO) developed the ISO 9000 standards [89]. These standards were designed to relieve organizations of the burden of producing a multitude of different versions of documentation of their quality system, for different customers [68]. There was one series of standards for every type of enterprises. The only difference was with ISO 9002 and ISO 9003. ISO 9002 was used in enterprises that had no formal research and development function within, ISO 9003 was used in enterprises without production. But isn't establishing a new organizational process in the company also a development? And some research has to be done before development starts. In 2000 a new standard came - a sequel, or better, a rewriting of the initial standard series from 1995. It remains one for all, but this time it is process oriented. TickIT [93], is not actually a

standard but more an application model for a standard, comprises a number of elements which augment the more general process of certification to ISO 9000 series standards and facilitates the achievement of quality in the software field [93]. This standard should be implemented in companies that design and develop software. It can also be used as an audit base for companies that purchase software from above mentioned enterprises. Standards do not guarantee high quality products; they only give us one of the possible ways to achieve traceability of our work. This is a way how to make the process transparent. Standards also do not prescribe exactly what kind of documentation should encircle the process. It only suggests what should be written down and who should do it and at which stage in the process.

Like ISO 9001 TickIT is divided into 20 subsections under a section called "Quality system requirements". These subsections represent the core of a documented quality system if a company decides to follow the standard. These subsections are [89]:

1. Management responsibility

   It is about commitment to quality. This represents a motto for the whole enterprise and must be well understood. Management must ensure that resources within the organization implement the enterprises motto and carry out regular inspections to check how it is being implemented.

2. Quality system

   The company must establish the system of writing down procedures that describe all the processes that are essential for ensuring the quality of the product.

3. Contract review

   When signing a new contract, everything that must be done, should be made clear. Contracts must ensure that in a legal and technical sense, customer requirements have been adequately defined and documented. The supplier must agree to these written demands and be able to comply with them.

4. Design control

   Quality is built into the product during the design stage. Procedures to review, verify and validate design should be written down so that the process is identical for every product and nothing is left out.

5. Document and data control

   This chapter describes possible ways to organize documentation. All documents that are part of a quality system should be approved and updated regularly.

6. Purchasing

   No enterprise produces everything needed in a production process itself. To ensure the high quality of a final product, component parts must be of a high quality too. This can be achieved with a regular evaluation of the suppliers, incoming control, in joined development with the supplier. In short, a procedure must be written down, whatever that procedure is.

7. Control of customer-supplied product

   Some of the parts we build into our product are supplied by our end-user. Unfortunately, we are even responsible for these, too. We are responsible for the quality of the whole product and should therefore establish a written procedure to inspect incoming contributions.

8. Product identification and traceability

   Traceability is one of the most important properties of the product and the process. One should be able to identify the origin of a product over any given time period. What tools we used, who performed what operation, what were the circumstances in which product was made, what kind of parts were used, etc.

9. Process control

   Processes that influence quality must be identified. A process must be carried out under controlled conditions that are: procedures documenting the manner of production, installation and servicing, suitable equipment, compliance to reference technical standards, monitoring of process parameters, approval of processes, suitable maintenance equipment.

   Standard ISO 9000: 2000 that replaced ISO 9000 series has process as the main stream of the standard. All other activities are connected to that stream.

10. Inspection and testing

    Although today's trends are to ensure and not to control quality, we can still not be totally on the safe side without testing activities to verify that the specified requirements for the product are met.

11. Control of inspection, measuring and test equipment

    To ensure the quality of the product, we must first ensure the quality of the measuring tools that control the process and product before, during and after the production stage. All measurements must be traceable to a higher standard and as such comparable to measurements done elsewhere. This requirement in the ISO standard is the easiest to audit and at the same time it is usually the most expensive to fulfill.

12. Inspection and test status

    With this procedure we must ensure that only the product that has passed the required inspections and tests is dispatched, used or installed.

13. Control of non-conforming product

    Products that do not conform to the specified requirements must not be installed. Procedures must allocate responsibility for such products to those who will ensure of their disposal.

14. Corrective and preventive action

    The supplier must establish procedures to carry out preventive and corrective actions. Customer complaints must be handled effectively and mistakes should not be repeated.

15. Handling, storage, packaging, preservation and delivery

    The method of handling, storing, packaging, preservation and delivery must be established, approved by the customer and regularly updated if praxis shows the need for that.

16. Control of quality records

    Quality records that evolve throughout the process must be identified, collected and kept. These records are part of product traceability.

17. Internal quality audits

    Internal audits should be planned and carried out regularly. They help us establish internally if our quality system is being implemented as planned.

18. Training

    The training needs for the employees must be identified, planned and carried out. Records about training given must be kept.

19. Servicing

    If servicing is a specified requirement, we must document the procedures to perform, verify and report that the servicing meets the specified requirements.

20. Statistical techniques

    If needed and used, statistical techniques must be written down and followed.

ISO/IEC 12207 - Information technology - Software life cycle processes

This standard [82] introduces three different parties into a software development business:

1. Acquirer
2. Supplier
3. Developer

It also introduces "software life cycle" processes into three groups of activities:

1. five primary processes
2. eight supporting processes
3. four organizational processes

This standard [82] clearly states in an acquisition process, which is one of the primary life cycle processes, that the acquirer will define and analyze the system requirements.

Being a public enterprise ELES must follow the Public Procurement Act [94]. The law controls every order in a form that is laden with documentation, formal demands and prescribed shape. However, the law does not give any ground that the tender will yield in a quality supplier and a quality product. This is understandable if we accept that the law catches many sorts of possibly acquired products and services.

As by law and so in the standard, it is prescribed that all demands for the product must be known before the supplier or developer is chosen. How this should be done is not prescribed and here lies the core of the problem. The solution (principle, mode, form) of the above should accelerate and cheapen the processes for the supplier (developer) and for the acquirer.

ISO/IEC 15846 - Software life cycle processes - Configuration management

This standard [83] is familiar with the term acquirer as it is introduced in the core standard for software life cycle processes ISO/IEC 12207.

ISO/IEC 15910 - Software user documentation process [84]

When preparing a project plan we often forget to plan time to produce working documentation for custom made software. Or, we make a mistake and leave it to the programmer to produce that documentation. It is my

opinion that documentation should be the result of the work of the analyst who will inform the programmer what the users want or need.

If we need to link standards to project management, it prescribes a project team, but this team deals with preparing documentation only. The team gets its responsibilities and the formal recognition of their work by being formally named to the project.

The standard describes everything about the outlook of the document; all the way to the size of the font, precisely how documentation should look like and estimates of the time required executing the task correctly.

When using specific standards some proportions of the time are prescribed for the production of software documentation [86]:

1. 15% for planning
2. 50% for the first draft
3. 25% for the second draft
4. 10% for the camera-ready copy

ELES prescribes how documentation should look like in its ISO quality manual. Instructions for using custom-made software are treated as all other working instructions. Authors only use a more graphical way to represent different screens that appear for the user. This is the easiest way to explain to the user what should happen after some keyboard action.

ISO 9294 - Information technology - Guidelines for the management of software documentation

This commitment requires recognition that software documentation is important and that it must be planned, written, reviewed, approved, produced, distributed and maintained [80].

ELES often regards documentation as an obligation towards the quality manual, it does not use it as a communication tool to the management and it never does find its purpose in task to task communication. Programmers are obligated to write documentation and other functions in software development such as analysts, subject area specialists, quality assurance specialists and auditors are not recognized. A role of a designer is included in the role of a programmer. Regrettably, ELES has not yet found the use of documentation as a historical reference to a project.

The only documentation maintained (as already mentioned) are working instructions. ELES has only now introduced formal project documentation such as project organization, time diagrams (Gantt-chart, PERT-nets), project charter (Work Breakdown Structure - WBS), nomination for the project, cost plan, etc.

It remains a task for the project office of the company to improve the understanding of project documentation, with the software development personnel and with outsourced companies.

ISO 6592 - Information technology - Guidelines for the documentation of computer-based application systems

To describe this standard [79] one must think of a quality system in a company even when it is not written down into a quality manual. If an auditor goes through such a system, he could determine, that a company demonstrates a fulfillment of the demands as they are written down in a quality standard like ISO 9001. People in the company just have not put that process down on paper. But in order to function, they must follow unwritten rules that can be documented.

As described in the case of a quality system, this standard does prescribe every item that the documentation of a computer-based application should include. Every item gets its own "box" in a table.

In everyday documentation we usually describe a couple of those boxes in a single sentence.

RTCA/DO-178B - Software Consideration in Airborne Systems and Equipment Certification

In this standard [92], software is viewed as part of a wider system or equipment configuration and the primary focus is on certifying that system or equipment. Six software processes (planning, development, verification, configuration management, quality assurance and certification liaison) are defined and various objectives and outputs are identified for each process [65].

ISO/IEC 20000 - Information technology - Service management

The standard [85] was issued in November 2005. It defines the requirements for a service provider to deliver managed services of an acceptable quality for its customers. In Figure 2.3 the closely related service management processes are shown that correspond to those that are described in guidelines ITIL (see also 2.2.4).

Other SW standards

There are many other current standards that are not separately listed in this work. Many of them are applicable in fields such as the defense, space industry, aircraft industry, medicine and pharmaceutical industries. In these cases, there is something in common. If something in software goes wrong, lives are at stake. The software mentioned in these standards is usually an integral part of a device and the correct functioning of this device depends

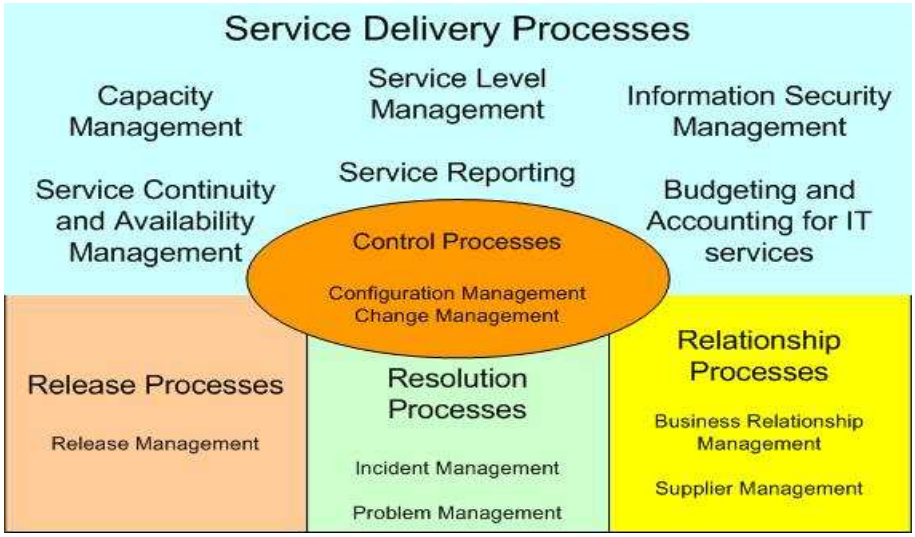

Figure 2.3: Service management processes

on correct functioning on it. To produce and test such software takes a large amount of money and time.

Article by Brezavšček and Zupan [11] takes special consideration in information security (standards and recommendations are mentioned). Importance of security in IT grows with the growth of Internet usage.

2.3.1 Using standards in a small enterprise

Small enterprise in this context does not refer to the total number of employees or a company's revenue, but to the size of IT staff whose job is supporting software in an enterprise whose core business is far from software development. Software in such cases is used to support processes in retail, finance, accounting, planning of work and finance, warehouse operations, business analysis, human resource management, inventory processes, etc. Development of such software takes place partly with in-house team members and partly by sub-contracting companies.

There are two ways of acquiring the applications for such a purpose. One is to buy and adopt software packages like SAP R3, BAAN, SCALA, etc. In such a case the main stress is on how to close the gap between the bought application and the process currently running in the enterprise in question. The question of standards is "How to standardize work?" Standards depend on the type of work that is done and have nothing to do with the software itself. The other possibility is to develop own custom made applications, where the applications conform to the processes that go on in an enterprise. The end-users of the applications are usually known by their names and

without wishing to patronize them, they “cannot be standardized" because each one is something special. This is the first rule one has to accept when producing custom made software in circumstances described here.

IT staff in such a company that does not use standards as described above, mainly because there is no time for that when software is being produced ad-hoc, and the process that is being supported is not standardized. Usually the only thing that has a standard form and content is the user's documentation, simply because users demand it. Companies that perform the outsourced work usually use internal standards adopted with their enterprise, because in many cases such companies are small with no more than 20 people and there is no time to study official standards other than those sometimes mentioned by government laws.

2.3.2 Documentation for applications users

Different profiles of users

A working application is the most important goal for every user. Documented requirements, models of statistical elements in the system, models of interactions, and other documentation about the problems that the programmers are working on are useful but only secondary. A customer placing an order will not be satisfied with a pile of documentation, if he or she does not see the working application first. The waterfall model[2] is the most predictive of the methodologies, stepping through requirements capture, analysis, design, coding, and testing in a strict, pre-planned sequence. Progress is generally measured in terms of deliverable artifacts - requirement specifications, design documents, test plans, code reviews, etc. One of the criticisms of the Waterfall model (see Figure 2.4) is that users must wait until the end of the project to get every component of the ordered software. All he/she can get in the meantime is the documentation. On the other hand documentation is of fundamental importance because it facilitates the maintenance of software, communication between programmers and users, provides a study of quality of work for the project group, etc. That is why it is important to get corresponding documentation. Every piece of documentation must be grounded [4]. Nowadays it is used for small stable projects.

Developers - Designers - Internal standards

The phrase “technical documentation” defines the documentation used by developers. It should in the first place provide **traceability**. Applications are usually developed by more then one developer even if they are not technically demanding. Often there will be other developers that take over the application once it is installed and maintained.

[2] Definition from http://en.wikipedia.org

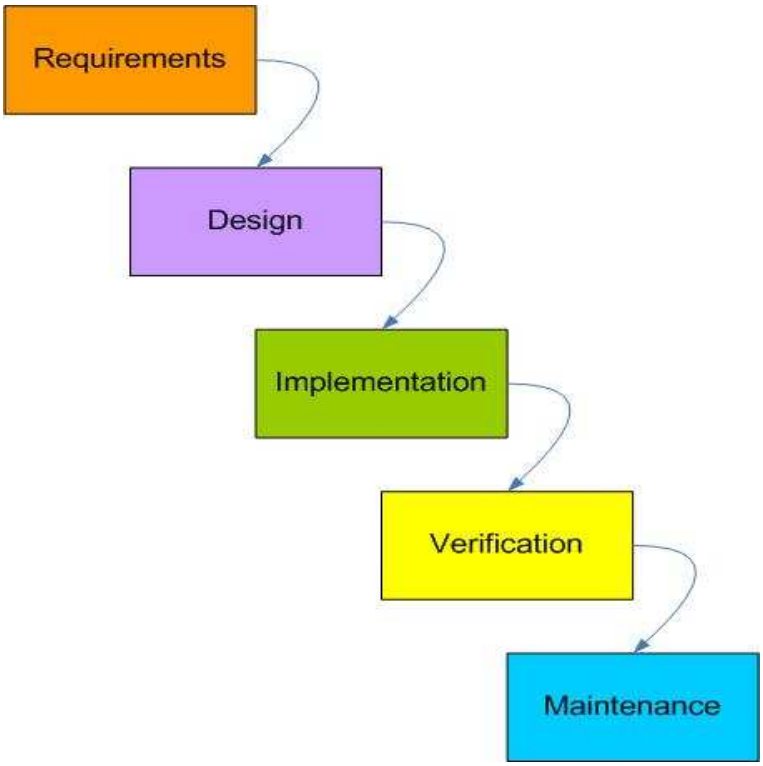

Figure 2.4: The Waterfall Model of software development

The technical documentation of an application should give the basic idea about entity relationships (ER diagrams), attributes, tables, views, forms and reports used in that application. With the help of such documentation even a new programmer on the team can quickly become familiar with the application he is developing, upgrading or correcting.

Internal standards in an enterprise for developing and coding new applications can also be of vital meaning.

End users of applications

End users of applications are employees who use computers to do their tasks. Applications should facilitate their tasks and documentation such as users' guidelines must help them in doing that. Applications can change on an almost daily basis in accordance with the wishes of users. Documentation cannot follow in the same pace. People are visual beings, and that is why documentation for end users contains pictures of Graphical Users Interface (GUIs) and actions are described as to what happens when a certain button is pressed, or a certain piece of data is input. Users do not need to know how a program works, but they should know their tasks and the consequences if not

performing them correctly. Documentation should provide some descriptions of how their work in the IT system influences others.

2.4 Public procurement as the first filter

Outsourcing in a public enterprise must be selected according to the Public Procurement Act (Public Procurement Law 2000) [94]. When taking this act into consideration we have four different ways of choosing the sub-contractor for the outsourced work, depending on how much money such co-operation is based on. For maintenance, usually the more expensive outsourcing comes into consideration (above 24.000 EUR). We have to acquire at least two complete offers. All documents that must be present in the complete offer are listed in written instructions available to all possible suppliers and are the same, no matter what is being purchased. In advance, criteria for selecting the contractor must be known and should be strictly followed when opening bids and choosing the future team co-workers from the sub-contractor. When the project team has completed the task of choosing an enterprise to which we shall subcontract some of the work on project, the next phase in the project can start.

2.5 Quality of organization

The quality of an organization is shown by the satisfaction of its employees [44] and consequently in the performance of the enterprise. The can enterprise be successful in the short term without employee satisfaction and the same goes vice versa. But only if satisfaction goes hand in hand with successfulness of the enterprise, can enterprises prosper. Good organizational quality does not necessarily give a quality product, but for a good product a quality organization is needed. A project is an organization in smaller proportions, which needs some services from a larger organization where the project is nested. It would be presumed that a good project can only come from a good quality organization. These two entities are connected by the services that one offers to the other. But is it enough to connect them without compromise? In a project things are more complicated although in smaller proportions, because at least in an IT project, a smaller number of people is involved. A project is really an organization within an organization. It is the product of a broader organization within the company and must produce a quality product of its own.

2.5.1 Influences on organizational quality

Employees

The task of the manager of an enterprise is to pick the best people for the purpose of an enterprise - making money and serving customers. It is not enough just to choose good programmers to have a good quality IT project team. We need much more than good programmers in an IT project team, programmers must also be congruent with the rest of the team.

Environment

The environment is the outside surrounding the company. Although programmers are supposed to be very introverted people, they must interact with their surroundings to gather information, ask questions, deliver the modules and finally bring the product to life.

Market

It is very important what kind of market our product is being aimed at. As mentioned in this work earlier, we differentiate between the companies that produce software for customers both inside or outside the enterprise. Linking the organization to the customer's business is essential [3]. This is the reason why we must know our customer well and have his wishes well defined. Possible ways of achieving this are described in the section about Quality Function Deployment. An IT company (or just a department) can have a customer that is highly IT aware, knows nothing about software except that he needs something. There are of course products made for mass use (Microsoft Office, CorelDraw,...) and the customer's own in-house developed software products. The latter is not the focus of this work.

2.5.2 Organizational culture

Takeishi [63] refers to the importance of organizational culture by pointing out the vital role played by powerful project leaders in cross-functional, intra-organizational and inter-organizational (outsourcing) coordination and problem solving. This strategy worked at some auto-makers, but not at those with "traditional values", where project leaders could not yield sufficient power.

2.6 Organizational relationships

Since the era of organizations was first proclaimed, new forms of organizations and ways of organizing seem to be emerging with increasing intensity.

Organizational phenomena are changing in the same way as our research perspectives. The creation of ever larger bureaucratic organizations goes hand in hand with the appearance of new, informal and network-like organizations.

For the purpose of this thesis we first have to differentiate between internal and external relationships [44] that are developed between employees or between employees and outside partners. The study of outside relationships was so far less absorbed and has a shorter history. For the time being, outside relationships will be represented by relationships between in-house employees and employees of the sub-contractor. Rosabeth Moss Kanter [47] said, that relationships between people and systems are as alive, as relationships between strategic partners are alive. They have numerous possibilities. Companies that are able to master those possibilities and use alliances to their advantages can strengthen their value in assets/investments.

Relationships are formed, they change and disappear. We see dynamic changes in relationships. The changes that grasp all the changing needs, interests and goals of the employees in a company reflect the quality of an organization and its ability to adopt.

There are many classifications of relationships. However, organizational relationships originate in "working orders" and are as follows:

- relationships of a technical or technological nature,
- relationships of a personnel nature,
- coordinative relationships,
- communicational relationships,
- motivational relationships.

Apart from the five relationships above we also have combined relationships where two of the basic ones are combined. Authors of the research described in [43] represent these relationships in the form of a bunch of grapes shown in Figure 2.5. This Figure is the result of research performed in the late 1980s by a group of Slovenian experts [43] that defined a methodology to assess the quality of organization[3]. We shall use this Figure later to show how is the ideal organization for achieving quality software solutions reflected in this structure.

2.7 Quality function deployment

Quality Function Deployment (QFD), also known as The House of Quality, is a device created to tie product and service design decisions directly to the

[3](MUKOZ) - Metoda za Ugotavljanje Kakovosti Organizacije Združb; Methodology of assessing the quality of firms' organization)

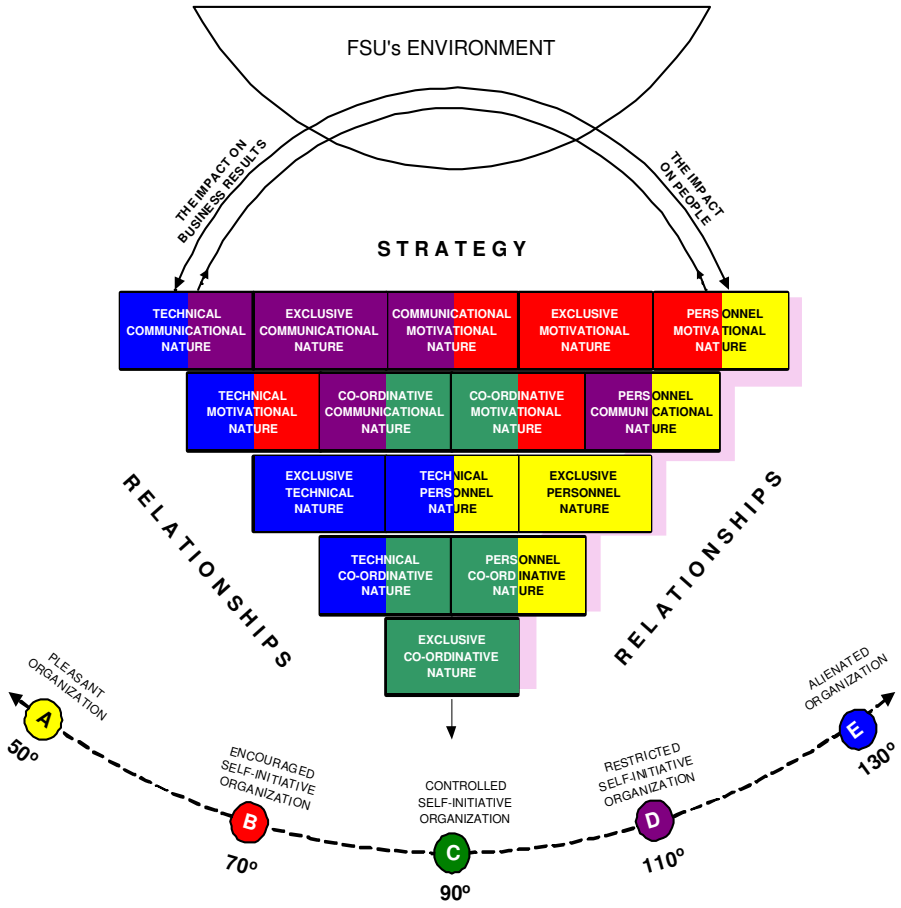

Figure 2.5: Organizational structure of relationships as a bunch of grapes [44]

customer wants and needs, and has led to a host of successful breakthrough products. QFD is designed to deploy customer input throughout the design, production, marketing, and delivery facets of a given product or service. In a typical QFD application, a cross-functional team creates and analyzes a matrix linking customer wants and needs to a set of product and service design metrics that the company can then measure and control.

Originally developed by a Japanese shipbuilding firm in the early 1970's, QFD was imported by the US auto industry in the 1980's and has now achieved widespread use throughout the world in just about every industry imaginable - both manufactured products as well as services [75].

2.7.1 Voice of the customer

What does the customer really want? As a producer or service provider one should focus on the customer and his/her true needs. It is important to acquire information from customers and to help them understand their needs.

One of the possibilities to achieve this is to complete the following table in Figure 2.6 together with the customer.

Table: Voice of the costumer		Reference	Internal/External	Socio-Economical Group	**WHY?** Why do you need or want this product/service?	**WHAT?** For what purpose will this product/service be used	**WHO?** Who uses the product/service now? Who will use it in the future?	**WHEN?** When is a product/service being used? When will it be used in the future?	**WHERE?** Where will the product/service be used?	**HOW?** How is the product/service being used or will be used?
Literal statements of the costumer		1.00								
		2.00								
		3.00								
		4.00								
		5.00								

Figure 2.6: Voice of the customer

Before we start to gather information from customers, we have to ask ourselves who the customers really are. In many cases there are several groups of customers. Usually we know the final customer, however there can be several buyers in between. As a case let us assume that we produce copy machines. The final customer is the one getting a copy out from the machine. In-between customers are repairmen that service the machines and a person that manages the machine. Every customer has its own group of demands that matters to him/her. [61]

2.7.2 From engineering to software engineering

Every service or product we provide has several internal customers when we speak about software. If software is really a product then when it is being used it provides a service. But what software developers have to take in to account is, that their product should offer services of high quality. Software can be of high quality, but if end users can not perform with it a good and quality service, the software does not meet its goal. Therefore software developers need to know not only the code, but also the service that will be offered with the code.

2.8 Quality assurance

The development of a quality orientation that uses internal processes to ensure the product can fulfill the stated specifications by focusing on prevention. Historically, it precedes total quality management that is an ideologically and culturally based system of managerial operation that seeks to improve continuously the total organizational system that produces goods and services to satisfy customers every time. When an organization engages in development, acquisition and/or maintenance of software, whether for profit (sale or resale) or as an internal service to the company, management must have a view of quality, and what it **should mean to them** in relation to the proposed software [7].

Quality assurance is a planned and systematic program of all actions necessary to provide adequate confidence that materials, data, supplies, and services conform to established technical requirements and achieve satisfactory performance.

A software quality assurance plan should consist of:

1. Planning and procedures.
2. Analysis of productivity and quality data.
3. Reviews, audits and inspections
4. Configuration management.
5. Software testing.
6. Specifications and documentation (See Section 2.3.2)

2.8.1 "Instant" Quality Assurance

In relation to quality of software, we are considering about small businesses that produce software for its own use with the help of outsourcing. Such software usually does not control people's lives or large technical systems (electricity management system, aircraft control, health management, etc.). Software developed in the way described in this work is used to manage business informatics within a company, such as technical data bases of different devices which are "not mission critical". Standards and procedures related to quality such as [89, 80, 74], prescribe a lot of paper work, both before and after work activity. In the companies described in Section 1.2 there are usually insufficient resources available for the job. Employees using programs developed "in-house" presume that the only job programmers have to do is to serve them and in a sense, they are right. It depends on the culture of the company if jobs necessary to ensure quality are regarded as useful and

are appreciated. Even if one gets appreciation, resources in people, time and money are still limited.

There are, in this situation, several "ways out" to ensure quality of software produced in-house with the help of outsourcing.

Automatization of documentation production

Technical documentation is necessary for quality software. Producing it is a painstaking job that nobody likes. However, technical documentation can be produced almost automatically by using appropriate "helping" software [6]. In Figure 2.7 a copy of a technical documentation that ELES produces is presented. This kind of technical documentation that especially new programmers use, is created automatically in the system. It gives them the idea tables, forms, triggers, attributes from which the application is created.

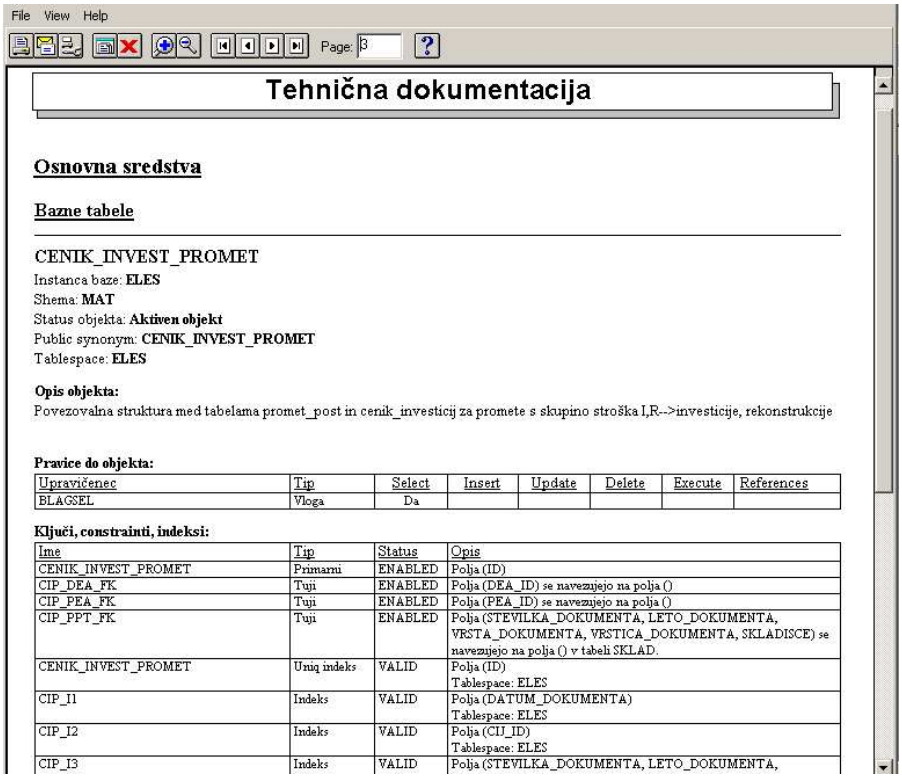

Figure 2.7: Appearance of technical documentation produced in ELES.

Educated and skilled users

Today, information technology, ERP (Enterprise Resource Planning) programs and personal computers are all part of everyday business in every

company in the developed world. Unfortunately, many employees (not only the older ones) cannot adapt to the changes going on for the past two decades.

Trust

Trust may concern a sub-contractors' ability to perform according to the intentions and expectations of a relationship, or his intentions not to defect willingly [51]. Trust is risk, and risk avoidance is the name of the game in business [25].

System Analysis and Design with UML

UML is an object-oriented modeling language used to describe information systems. It provides a common vocabulary of object-oriented terms and a set of diagramming techniques that are rich enough to model any system development project from analysis through implementation [16]. The UML is an industry-standard language that allows us to clearly communicate requirements, architectures and design [90]. One of the processes using UML as a tool is Rational Unified Process. Use-cases are the foundation of this approach to software development. They show how system interacts step by step with the actors and what the system does.

Information that can be found in different diagrams of UML is not enough to represent future application in scope that will give us all the factors needed to calculate time, resources and cost needed to complete the project.

Chapter 3

Project Management

The function of every day mastering and focusing of the work of employees in a company in accordance with the given goals of the owners, that is performed by a hired manager is called the function of management [44]. Every manager has roughly three tasks:

1. Receiving "task and authority" to perform tasks from the company's board who's performing and trusted agent he or she is,

2. To carry out those tasks with the help of other employees in phases of planning, execution and control,

3. Ensuring that tasks, separated due to technical division of work, stay part of the business process.

Or in a more common language; the manager has two jobs:

1. Providing resources to run the business,

2. Building and maintaining a high quality organizational structure to support the business process.

A management activity that introduces a new objective or causes change and has a definite start and finish time is a project. It is a unique set of co-ordinated activities, with definite starting and terminating points, undertaken by an individual or an enterprise to meet specific objectives within a defined schedule, cost and performance parameters.

Projects bring added value to the enterprise only when line organization can not perform the work. Business functions should provide resources for different projects (see Figure 3.1 [45]; p.14).

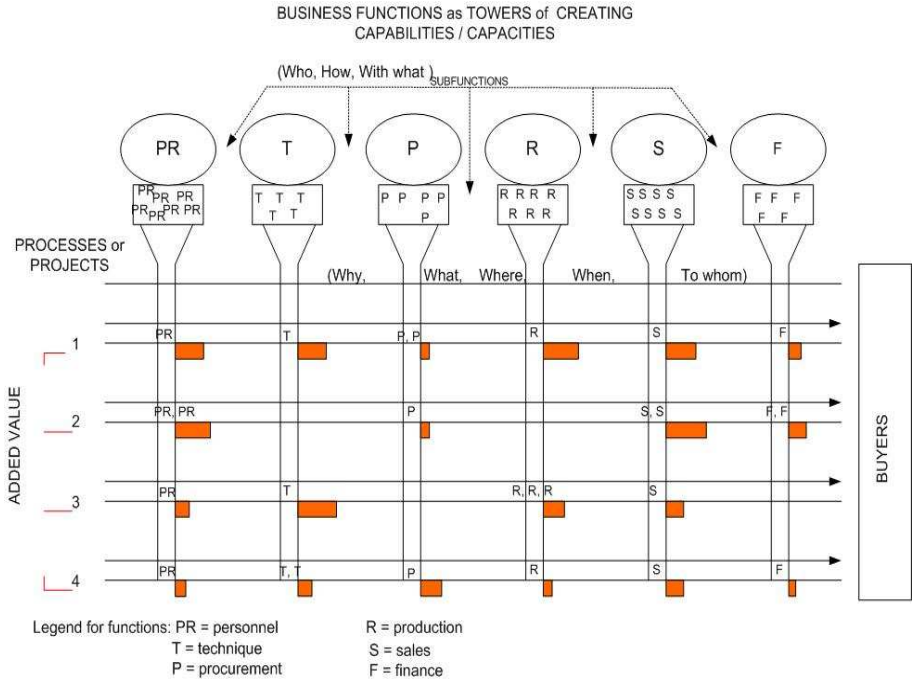

Figure 3.1: Business function providing resources for projects to create added value [45]

3.1 Coordination of work in projects

Software is usually produced in project groups. To co-ordinate the work done as a group, participants have to communicate with each other. Leaders of the groups should therefore be acquainted with the psychological and sociological principles of group work [58].

Projects are in literature [49, 73] limited by three interacting factors:

1. time,
2. resources,
3. costs.

A fourth factor connected with all three mentioned above is quality. In the theory of project management, the last one is mentioned more frequently in later period. A customer usually sets at least two factors and then presumes that high quality will just come as standard. To achieve the time limit and costs that a customer is willing to pay, we have to put more resources on the job.

Project management standards and experiences prescribe techniques how to organize the group, how to lead people in the group and how to achieve traceability of a project. A project has its own life cycle, different influences and "leading actors" during its life phases. Figure 3.2 shows the usual activities in the project during its time of existence.

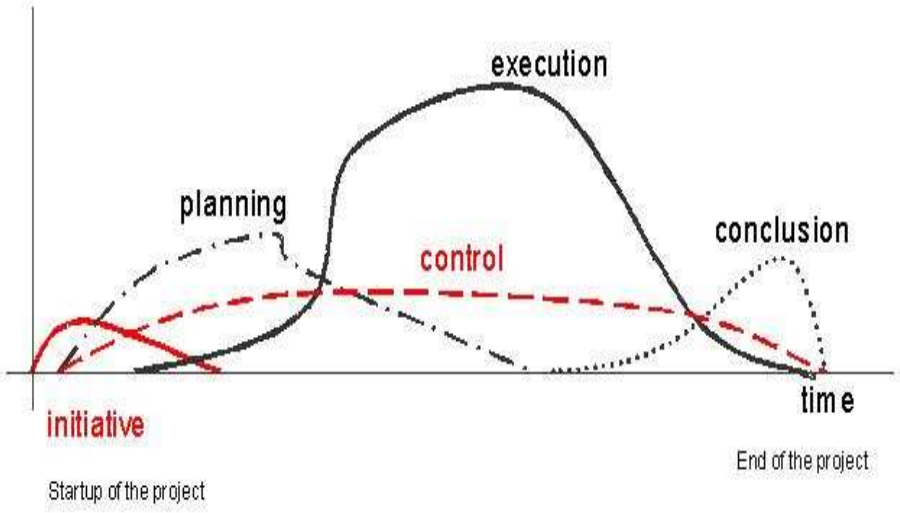

Figure 3.2: Project life cycle on a time scale

3.1.1 Organizational structure in an enterprise

IT projects are usually produced in at least two companies. One is the company that shall use the software solution as part of its every day activities, the other one actually produces or "codes" the software. The first one may or may not be a project oriented company. The core business of this company is not necessarily connected with software. It only uses software as a tool. However it is prudent to fasten together the core process and the project in the IT field. See Figure 3.3 where project phases are presented and their connection with the rest of the enterprises processes. Planning of the project has to clear some features with the enterprises management, execution of the project needs employees from the enterprise.

For the software producer, production is usually organized by projects. Every customer has its own project or maybe projects of applications produced are used in different fields. Every project has at the sub-contractor's company its own dedicated project manager. Not to mention that each project has a project manager also within the enterprise for whom the software is being produced. We are actually dealing with two people doing the same job. The first one has a much narrower job description. His job is to

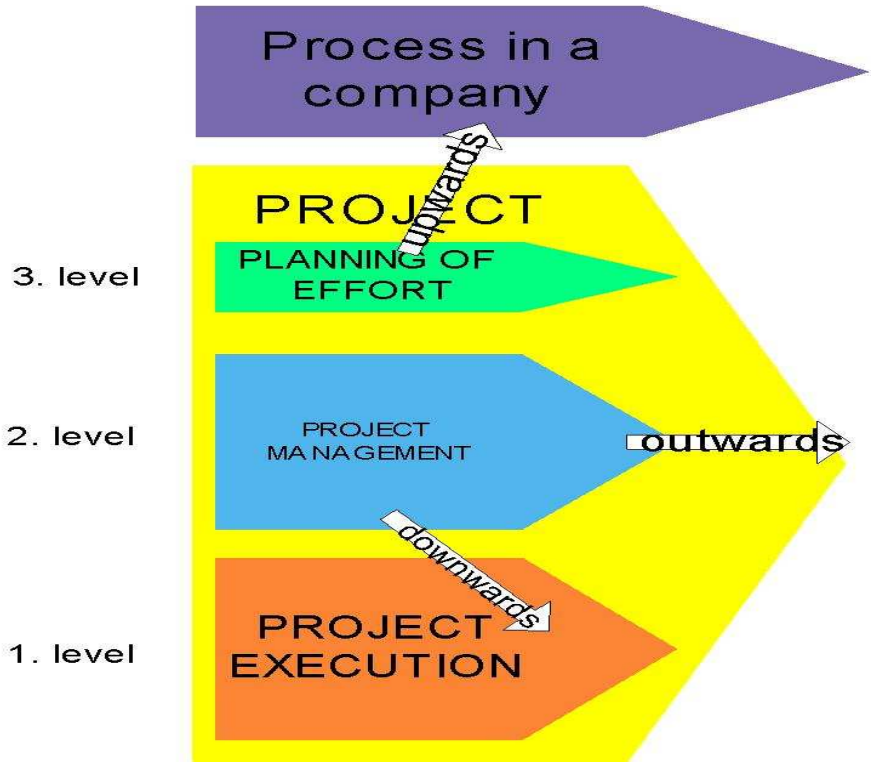

Figure 3.3: Possible way to fasten a project to the core process of a non project oriented organization

manage programmers in the sub-contracting company and to get the customers system expectation in a form he/she can translate into software. If the specification is not correct or not complete it is really not sub-contractor's problem. Sub7contracting company can just charge buyer money that was originally agreed and then charge buyer on top for the requested changes. The project manager in the company that's buying the software has a much broader description of his/hers responsibilities. He or she is the one responsible for the implementation of the software and for the success of the entire project. So the manager in the company for which the software is developed is "the real" project manager.

3.1.2 Roles in a project team

Project manager

The work of a manager is described in theory by Henry Fayol (1841-1925) [44] with this main activities:

1. Planning,

2. Organizing,
3. Ordering,
4. Controlling,
5. Coordinating or Reconciliation.

All these activities are also vital for a project manager, but are they enough? A project manager must possess skills and knowledge. Knowledge can be obtained; skills are often "God given". The third property that is necessary for a successful project manager is delegated competences. With this instrument he or she should accept responsibility and obtain power to manage the project. In a company where software only supports the core businesses, software projects are of secondary importance. Chief process owners (CPO) are at this time not named. During the next five years, the process owner role will become critical in ensuring ongoing process improvement and integration [88]. But for now, the project manager is on his own. When choosing the subcontractor for programming in a state owned enterprise he/she is limited by the Public Procurement Act, and usually he/she cannot choose internal staff to be named onto the project team as functional leaders of other divisions will often resist such move.

Organizations must recognize the importance of behavioral factors in working relationships. When they do, they come to understand that project managers should be hired for their overall project management competency, not for their technical knowledge alone. Today, project managers are more managers of people than they are managers of technology [24]. In [17] authors list 8 items that differentiate best performing project manager (leader of the project team) from an average performer:

1. Pattern Recognition: The ability to identify key issues in complex situations, to condense large amount of data into useful form, etc.
2. Use of Concepts: The ability to apply concepts and principles, to draw logical conclusions, etc.
3. Use of Influence Strategies: To develop sequences of actions, communicating, negotiating and using experts or third parties to influence others or to shape a situation according to one's desire, etc.
4. Achievement: The desire to achieve, to commit oneself to accomplishing challenging objectives, to compete against a self-defined standard of excellence, taking calculated risks, etc.
5. Initiative: Being self directed, proactive, taking action before being asked or required to, etc.

6. Time Management: The "internal clock" that organizes time frames to make things happen, etc.
7. Group Management: Building teamwork while leading a group, managing conflict, maintains resolutions, etc.
8. Self Confidence: A belief in one's capability to live on the border line, to cope with stressful situations, to maintain calm under pressure, being accountable for results mainly dependent on others.

Above listed items or person's characteristics are in different form repeated at the end of thesis. Some additional characteristic will be added. Listed factors that characterize a good project leader are also important in other team members.

Sponsor

A sponsor or sponsors are people who want to see the project succeed. They want a positive results or outcome of the project work. If they have a position and reputation within a company or they are involved in a broader project, they will not want that project to be endangered.

Project staff

In a non-project organization, project work is considered obsolete. If a project manager could choose the best people for a project, from the functional departments, he would look for those that are deeply conscientious about their responsibility within a project. They are often not paid for extra project work. In situations when one cannot choose project staff, and software development projects are not popular, one offend gets employees that in their own department do not have a lot of work. "Why?" is the question that it is not polite to ask.

3.1.3 How to choose team members

Who shall form the project team is the question we have to answer before the project start. The team can be formed through agreements or team members can be simply appointed for the job. There are techniques like Moren's sociometry or the technique of a mandator. Where possible team members chose their coworkers in a team. However none of those techniques can predict the success of the team or a quality of the result. Adizes [38] proposes that a good team is formed out of complementary people that supplement each other. For him a starting-point for choosing a team is behavior patterns that candidates have. Out of patterns different roles in a team are derived from. In theory if all roles are taken, the team is successful. Later in this work I shall argue that usually one can not choose ideal team

members neither team members are tested for the role they can play in a team. Practice in an environment described all through this thesis showed that other factors are important and give us some possible prediction about the success of the project.

3.2 Different project cultures within the organization

Do different perspectives on projects depend on education, age or is it simply a way of thinking about one's own environment?

We find several different professions in a company, which represent the case in this thesis. Electro engineers and electro technicians are the most common. They are from 25 up to 65 years old and usually male. Other professions are sociologists, civil engineers, lawyers, economists and the rest are a variety of other professions. Almost 50% of all employees have a high education. Education does not only gives you the knowledge, it also gives the way of thinking too [32].

One's profession, however, is not the only consideration that determines one's perception of projects and project management. Fundamental differences between departments are arising from the nature of the work. One's work is strictly processing, the others are builders and investors. On courses about cultural differences, project managers are taught how Germans think, what Italians understand and why Americans do it their own way. In every day life, project managers may encounter even more distant cultures within one's company. Up to some point, a project manager can be educated about how to deal with a situation when cultures collide; the rest is up to the project manager him/herself.

3.3 IT projects "hot spots"

From over 40 research articles which have investigated knowledge practices within a project context, a model was developed [53] to show where the application of knowledge management principles might affect IT project success [54]. Figure 3.4 contains a simplified model of an IT project, with Inputs, Processes, and Outputs shown. Within the project, there is a Governance process and four typical project phases - Plan, Design, Build, and Implement. Superimposed on this model are the ten "hot spots" identified in the literature. Each hot spot will be discussed below.

Project Inputs

There are two issues to consider at the beginning of each project - team selection (hot spot 1), and timing of entry and exit of team members (hot

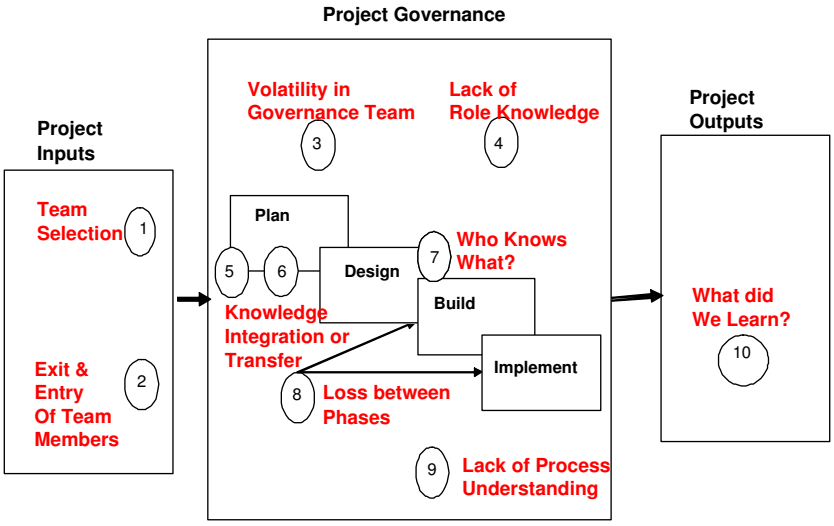

Figure 3.4: "Hot spots" of IT projects [53]

spot 2). Team selection is important since the project manager will want all relevant knowledge areas included in the team or available to the team when needed. Problems arise when the project manager cannot select his/her team or when the knowledge profiles of team members are not available during the selection process.

The second "hot spot" is the timing of entry and exit for team members. Walz et al. [69] have shown that members added to a team after the midpoint rarely change the team's direction. In addition, loss of team members will result in knowledge gaps within the project.

Project Governance

Project governance can include the roles of executive sponsor, project champion, project manager, and project steering committee. From a knowledge management perspective, two issues are identified - volatility in membership (hot spot 3) and role understanding. After a project begins and the governance structures are put in place, there is a gradual knowledge building process among the key stake-holders. When members of the governance team change, there is the distinct possibility of a knowledge gap, especially when replacement members begin to influence the team. This volatility is most important with the project manager but extends to any member of the governance structure who controls or influences resources that the project needs.

The other issue in governance is lack of role knowledge among the governance team (hot spot 4). When senior executives take on project sponsor or

champion roles, they do so because the outcome of the project is important to them personally and to the organization. Unfortunately, there is often no training given for these roles. A first-time project sponsor may not know when to support the project and when to tighten up the reins; or whether project problems are serious or temporal. Gaps in project governance knowledge endanger the success of projects.

Project Phases - Plan, Design, Build and Implement

There are five "hot spots" within the main body of an IT project. Two have been identified in the Plan phase, which is the most highly researched project phase, probably because of the high cost of errors or omissions in this phase. Within a project which is building a custom application, knowledge integration between users and analysts is critical (hot spot 5). Within a project which is implementing packaged software, knowledge transfer from vendor or consultant to the internal project members is critical (hot spot 6). In both of these cases, there are often tasks on the project plan for these knowledge-related tasks to take place, but no objective way to measure how effective they were. Failures in this phase result in rework and add to the expense of a project.

Within the Design and Build phases of an IT project, there are a multitude of interrelated decisions to be made. The more innovative the project is with respect to technology, the more important it is that team members know "who knows what (hot spot 7)", so that design and build problems are addressed by the best knowledge source in an efficient manner.

Because team composition changes from phase to phase in IT projects, knowledge loss between phases, "hot spot" number (8), is particularly important. There is a significant risk that the knowledge generated by one phase will be inadequately transmitted to the next phase. Traditional methods of documentation, such as models, state diagrams, and use cases, rarely capture the "why" of design choices.

Within each phase, "hot spot" number (9) (Lack of Process understanding), there is the need for project participants to have an understanding of the project structure and process, and where they and others fit in. Without this knowledge, team members can waste time on low priority tasks and will fail to coordinate with others whose work depends on theirs.

Project Outputs

As organizations become more reliant on projects for transformation, renewal and competitive advantage, the outcome of a particular project may be less important than the overall increase in the ability of an organization to implement projects successfully. This means that knowledge captured at the end of a project is potentially very important to the enterprise as a whole.

Research has shown that post project reviews, if held, must be broadly attended, and that the transmission of "lessons learned (hot spot 10)" from these meetings can be problematic.

3.4 Large IT project

Most enterprises whose core business is coding software are involved into large projects. Sometimes those projects have up to 100 team members. Team members are divided into different sub-teams like:

- Project management; responsible to manage the project and communicate with the customer.
- Development team; responsible to develop a product. Sub teams can be responsible for development of different product modules. Team members are responsible for writing technical specifications, coding, writing technical documentation, unit testing, reviewing test plans and plans for user's documentation.
- Team for writing users documentation.
- Configuration management team.
- Quality assurance team.
- Customer support team.

3.4.1 Costs

When running a project of this size, a lot of time is spent on software estimation process [60]. Project managers use different techniques from COCOMO (COnstructive COst MOdel) to Used Cases. The model must be tuned in the company culture - otherwise it is not useful. Specifications must be written down thoroughly, there are team members in charge to check them. Whenever estimating work to be done, number of people needed to do it, cost and quality of the final product, unknown variables are often dependent on people on the project and how they can do the work in a team (team structure, experience of team members, style of team work, etc.).

Model COCOMO II uses the algorithm to determine the nominal time (PM_{NS}) needed to perform the work in Engineer Month units. There are several factors influencing the result (*EMi*; Effort Multipliers, *Size*; size of the project, A; constant), but for the purpose of this thesis we shall look into one of them with more detail.

$$PM_{NS} = A \times Size^{E} \times \prod_{i=1}^{n} EM_i$$

Input parameters of the model COCOMO II

Factor E depends on several other variables and defines the economics of the project. When E is smaller then 1, productivity grows. E becomes larger usually for two reasons (see [60]). Communication between team members becomes more important and additional work is needed when integrating large systems. There are 5 factors defining E:

1. Knowledge about past projects (PREC - Precendentedness),
2. Development Flexibility (FLEX),
3. Architecture for Risk Resolution (RESL),
4. Team Cohesion (TEAM),
5. Process Maturity (PMAT).

My interest is only in factor *TEAM* . When defining *TEAM*, we have to consider:

- Consistency of groups emerging from software development (buyers, developers, testing team, maintenance team,...),
- Readiness to conform between groups,
- Experiences at team work,
- Associating of groups, that enables joint vision and higher synergy.

People and their behavior definitely influence the outcome of the project.

In case of this thesis we would use the organic mode of development as described by the COCOMO author Barry Boehm. This is the mode where relatively small software teams develop software in a highly familiar, in-house environment. Most people connected with the project have extensive experience in working with related systems within the organization, have a thorough understanding of how the system under development will contribute to the organizations objectives.

COCOMO II model does not consider quality. Costs are not directly related to the quality of the final product.

3.4.2 Scope

Beside costs, scope of software development project and with it time needed to complete given tasks is the other important factor in large IT projects. In practice the deadline for conclusion is usually given by users - buyers. Author in [72] introduces and estimates some of the methods used to asses project scope - Functional Size Measurement (FSM). The foundation for his work is UML as a standard. Some of the possible approaches are:

1. Function Point Analyses (FPA); main positive characteristics of the method are independency from used technology,

2. Future Points,

3. Full Function Points and others.

With the above mentioned methods the project team tries to evaluate the scope and the effort needed to complete a project. Time as one side of the project triangle is in focus here and so are the resources needed. Boundaries of the project should be set before the start. To a given problem one tries to prescribe a number that shall tell us in empirical way what number of hours and programmers shall be needed. Applications are formed out of Transaction types (T) and Data group types (F) and program system (H) is then represented as a vector - mathematical record of the planned system;

$$H = (T_1, ..., T_\tau, F_1, ..., F_\tau)$$

Transaction types and Data group types can be further subdivided and other mathematical algorithms are used to get to the final number we are looking for, but I shall not go further in to details here. Neither the calculated time needed to finish the project nor the right amount of resources can provide us with the successfully completed information project that will be done within the cost amount planned and in quality demanded. In practice we are faced with additional problems. Specifications for the application developed are changing constantly and unpredictably.

3.5 Project information system

When a company manages projects as part of its everyday job, it also needs a project information system. This system should be designed in a way to provide management and project managers with up-to-date informations on projects and to give them the tools to make decisions concerning current and future projects. Project managers use project information systems to reach set goals on time and within budget. They do it through the following realization and in communication with their associates [64]. Projects have to be supported during the three phases of their life cycle:

1. Project start-up - design,

2. Preparation for execution,

3. Supervision of execution of the project.

Responsible managers and other participating employees have to be provided with information throughout the entire life cycle of the project. The most important data about a given project are time, money, human resources and quality. All this data enters the data base system via other business functions and are often not originated by the project itself.

Time

"Time is not the enemy; it is a mean to reach our goals." Network planning and network diagrams are something most people imagine when hearing a word "project". Those diagrams (Gantt and PERT charts) enable project planners to define tasks, their duration and the order in which they follow each other. In the start-up phase we plan for how much time will be necessary; later on in the project cycle we can only supervise how well we have spent it. A project manager plans time on the basis of his/hers experiences, empirical data and other similar projects. For some tasks, the exact time is hard or even impossible to determine.

Finance - expenses

Where do we get the money from and how wisely do we spend it on the project? Projects should have a financial plan and the project manager must be familiar with it. Contracts and orders that belong to the project start a financial flow outside the company. In return come receipts that stipulate when money is actually transferred to the business partner's account.

Human resources

People are the most important asset of any company and this applies to the project too. In companies like ELES which are not managed by projects, employees have, in the case when they are working on a project, two masters; a functional manager and a project manager.

3.6 Projects in organization and informatics

Organization and informatics go together and the informatics cannot be done correctly if it is not preceded by the organizational project. More than many other types of projects in engineering, mechanics, energetics, etc. IT projects touch the lives of people and the way they are used to work. They bring changes on micro levels.

The usual inspiration for project methodologies came from engineering disciplines such as civil or mechanical engineering. Such disciplines put a lot of emphasis on planning before you build. Such engineers will work on a series of drawings that precisely indicate what needs to be build and how

these things need to be put together. Many design decisions, such as how to deal with the load on a bridge, are made as the drawings are produced. The drawings are then handed over to a different group, often a different company, to be built. It's assumed that the construction process will follow the drawings. In the practice the constructors will run into some problems, but these are usually small. Since the drawings specify the pieces and how they need to be put together, they act as the foundation for a detailed construction plan. Such a plan can figure out the task that needs to be done and what dependencies exist between these tasks. This allows for reasonably predictable schedule and budget for construction. It also says in detail how the people doing the construction work should do their work. This allows the construction to be less skilled intellectually, although they are often very skilled manually [19]. Design and construction are two activities that are fundamentally different yet the second one is dependant on the first. Design requires expensive and creative people, construction should than be easier to predict. To be able to use people with lower skills in construction, we would have to have a straightforward output of design in the first phase.

In software engineering the design phase does not cost not about 10 % of the whole project but it is much more expensive. The costs of programming are much lower then the costs of construction when a bridge. The design in IT project takes at least 50 % of the project time. This brings us to some important differences between traditional projects and IT (or organizational) projects:

- In software: construction (coding) does not need a lot of material and other assets.
- It is in IT projects hard to distinguish between specification and design. Both phases can overlap.
- In software all the effort is design, and thus requires creative and talented people with broader perspective.
- Creative processes are not easily planned, and so predictability may well be an impossible target.
- We should be very skeptic of the traditional engineering metaphor for building software. It's a different kind of activity and requires a different process.

Chapter 4

Outsourcing

Outsourcing is defined as the management and/or day-to-day execution of an entire business function by a third party provider. A significant amount of management control is in this process transferred to the supplier. Outsourcing always involves a considerable degree of two-way information exchange, co-ordination, and trust[1].

Enterprises like ELES outsource part of their function via subcontracting services like software programming. The core business of ELES is to transfer energy on high voltage lines, to control the Slovenian electricity system and to organize the electricity market [77]. Developing software is not part of this core business. There are other companies that have software development as their core business and they (should) know their core business best. A subcontracting software company can therefore dedicate more resources to software development than their costumers or buyers for whom software development is not their core business. They have the experience, a library of code and they have done this many times so the knowledge they possess can save time.

In this thesis we study the following outsourcing scenario where the buyer and the subcontracting company specialized in software development form a joint project team. In this project team which is assembled for a specific software development task are members from both companies. This is not the case of mere "bodyleasing", where the subcontracting company leans a certain number of its employees to the buyer where they cooperate on projects which are entirely handed by the buyer (see [62]). This co-operation can evolve in a friendly relationship, stay strictly business based or become a hostile one. Outsourcing is complex and strategic and executives need a breadth of skills and a depth of knowledge to do it successfully [21]. Relationships are not just a result of one factor and cannot be expressed in one way only. They are a consequence of information shared between people and at the same time a source of new information. They cannot be built hastily, but over a given

[1] Definition from: http://wickipedia.org

period of time.

4.1 Outsourcing in the field of IT software development projects

Practice in IT department[2]

Most developers have experience with the corporate development environment. This is because there are so many organizations with developers. In reality there are two kinds of corporations that take part in development: large organizations with structure and processes, and smaller organizations who are trying to develop software with few resources [8]. For the purpose of this thesis I shall look into first of the two kinds of corporations mention above. In larger organizations, the development process is more defined, however not as defined as are software development processes in other type of organizations such as in software development companies. The larger the organization, the more systems are in production. In the larger organization, processing help desks are routinely routed to developers for investigation and resolution. This practice not only prevents the addition of new features, but also creates a great deal of distraction, which is never good for software development.

Managing IT the way it will give the enterprise a business value is faced with a difference in understanding what the role of IT should be. IT in an enterprise must serve the business and different process that run the business. In theory, IT should only do what is ordered from other processes. This is in practice unrealistic. Mangers in IT must have good relationship with other process managers and try to smoothly direct the development. In theory, IT also needs clear strategy and visualized goals of the enterprise. In practice, IT is the one that sets the pace. Processes in IT should be standardized, but compatibility of views is only achieved with good relationships between employees and other actors in the process of IT development.

The traditional view of outsourcing - letting a third-party run the entire IT operation - is currently the exception. Selective outsourcing has become the dominant model. Many IT organizations still outsource infrastructure or send applications to be produced in offshoring,[3] but others outsource specific functions such as, storage, help desk, or security. Business process outsourcing further complicates the market with additional options e.g. HR (human resources) outsourcing could include payroll, benefits, staffing, etc. IT outsourcing is significantly less mature than other outsourcing sectors

[2] In the thesis outsourcing of management of IT assets for the corporation is not discussed.

[3] "Offshoring" represents the transfer of an organizational function to another county, regardless of whether the work stays in corporation or not (Wikipedia.org)

(e.g. manufacturing), as it is lacking well-defined products, consistent market definitions and standard pricing [76].

Management in ELES usually involves outsourcing because ELES cannot manage software development projects itself. ELES gives the work involved in software development out to facilitate work in their IT department, and to mitigate risk. At least that is how it is perceived. It is never planned that if something should go very wrong that ELES would have to pull the work back in-house in case of need. In project management this would be the contingency plan.

Employees in the IT department serve as a support group for outside software development. These employees usually take care of the completed product during its usage phase and if necessary, they update it.

Why IT department?

The IT department was the flag carrier of IT development in ELES back in 1995. From the beginning the IT department had a very strong outsourcing policy not only for coding software but also for planning it. The future organization was therefore dictated from the outside. That has shown not to be the best possible solution. There was never a good snap-shot of the current situation made and foreign practices were introduced. That caused a revolt in the finance department. They revolted because their working practices changed and it took a long time for software changes to come through. The finance department was the first whose work depended heavily on computer applications.

Hence, it was necessary to begin supporting outsourcing in-house. Hardware had to be ready and running; not only servers but also the PCs on every desk where new software was introduced. Users had to be trained and new software had to be repaired, sometimes daily. When a program is being tested it is too late to make fundamental design changes! Design is the core process of software development that should not be done outside one's company. The key question is whether it should be done by the IT department alone or whether other employees from other departments should be included?

Management tools used to master the project

We are not aware that outsourcing requires even more skilful management than in-house development. A project manager inside ELES has to co-ordinate work with in-house employees and the project manager of the sub-contracting company. A situation often arises when the external project manager has to interview employees about the process within ELES. That can cause confusion with the employees and gives inaccurate results. Especially, if a project is unsuccessful and the same interviews with the staff

are repeated more often.

Since recently, the computer program MS Project is used for planning the project. Unfortunately, the program is not used as it should be also for tracking and updating the same project. ELES never demands nor gets voluntarily from their subcontractor companies such project plans. Companies that ELES engages for coding or additional tasks (analyzing processes, counseling) are not using any quality system. ISO 9000 series is not that popular with smaller software companies and the larger companies are too expensive to hire or are not interested at all to work for ELES.

The Project Office at ELES has prepared an organizational procedure that should be used for all projects in ELES. It prescribes responsibilities for the project from start-up till closure.

4.1.1 Major Risks

Some authors [42] breakdown the risks of outsourcing into the following items:

1. Transfer of expertise outside the organization,
2. Loss of control over future development,
3. Compromise of confidential information,
4. Loss of progress visibility and control.

The research done by Trestle Group [26] provides us with the results of recurring client issues in the field of IT outsourcing:

32% of interviewed customers say that "There is substantial gap between promises made in the sales process and the delivered results."

28% of interviewed customers say that "There is a concern about the proper handling of customer, people during/after the transition to the subcontractor."

24% of interviewed customers say that "A growing set of customers are looking for more business value from outsourcing, not just cost savings or personnel productivity."

36% of interviewed customers say that "Providers are not pro-active in suggesting new technology or business transformation ideas."

24% of interviewed customers say that "They would like their providers to help find ways to generate new knowledge to improve performance or gain competitive advantage."

Clear communication is needed between both parties, that can be achieved on a formal and on an informal relationship way. On the formal level one finds techniques used for relationship management, project management and milestones and adjustments. Informal ways of communication are relationship alignment, agendas and interests, priorities set. Both ways clearly state

that relationships are of high importance, managed formally or informally. Just building communicational structures is not enough.

4.1.2 Best practice in outsourcing

Reviewing the literature [52] about best practices in the field of outsourcing, one comes over the fact that relationships between two or more enterprises involved in outsourcing, are highly important. In long-term relationships where members of both parties' management teams have stayed in place throughout the duration of the relationship, a key component of success often cited is the development of peer friendships and in-depth understanding of one's counterpart in the other company. Suddenly, it is not important only how well the management structure can "close the deal" and what they put on paper. Important is if people involved go out on informal lunches or breakfasts, where they can catch up on all sorts of issues. When the subcontractor personnel come to the buyer's premises they should be treated as though they are part of buyer's enterprise staff. Weekly meetings are desirable but should not be forced.

Team building is something many project managers use when trying to get the work going. There are numerous techniques on how to do it. As the adage says that quality has a name, so, since quality of outsourcing depends on relationships formed in the team, it does have several names.

It is not possible to manage a project team composed of ELES employees and employees from the subcontracting company in a hierarchical way. There are typically two project managers, one that belongs to the enterprise and the other that comes from the subcontractor company. So the first rule that must be agreed on is that the project manager on the buyer's side has the last word. From that point on, all is in the relationship project managers can establish between them-self and towards other team members.

Chapter 5

Survey of project team relationships

In year 2003 I carried out a survey to get a feedback on how people that work for the sub-contracting companies and participate at software development for ELES, feel about working for ELES. The research tried to find out what are their concerns and how do they rate different circumstances in comparison with other companies for whom they have worked as sub-contractors in software development. Results of the survey should encourage ELES to stimulate team members to cooperate better, more often and on a different basis. The results of this research should identify which relationships are the most important in this cooperation and which relationships are deficient and should be improved. Similar research described in [70] inspired this research with the difference, that they were looking for key factors, which may have contributed to project failures.

5.1 Questionnaire

The questionnaire used in this survey was designed for the employees of sub-contracting companies who did software development for ELES. The questionnaire consists of 34 statements and was graded by 10 employees of three different sub-contracting companies. Statements were composed by the author of the thesis. The survey was done at the end or towards the end of successfully finished projects. The survey was anonymous. The only additional information that was gathered about those who answered it was how long they worked on different ELES's project (when they first worked for ELES) and what is the grade of their education. This additional information shows that they coopered with ELES on different projects from 2 to 8 years and that they are all highly educated. Because ELES has intensified it's software development project since 2001 we disregarded possible differences in answers from those who known and has cooperate with ELES for

a longer period of time. There was no substantial difference in education of the personnel who answered the survey.

Usually, it is the sellers who ask their buyers whether they are satisfied with their services, but this time it was vice versa. The buyer asked the sub-contractor to rate 34 statements in relation to ELES and in relation to the best rated company for each individual statement, among all other companies that they have worked for as a sub-contractor. The statements were mostly about how the employees of the subcontractor (no managerial staff was surveyed) were satisfied with the working conditions at ELES, with the management of the project, and with the relationship to the in-house project team members. The statements and grades are listed in Tables 5.1 and 5.2. Statements were rated from 1 to 5 (1 = totally disagree, 5 = totally agree).

Grades shown in the table were simply calculated as an arithmetical mean of the individual grades. In the analysis of the survey we were looking for the largest difference between the two grades.

5.2 Analysis

Most of ELES grades were close or above the grades of the best partner grade indicating that ELES is in general considered a good partner in the outsourcing relationship. We identified however six statements where the grades between ELES and the "best company" differ substantially. In the analysis of the grades we shall concentrate on those six issues.

Five statements where ELES was graded much lower in comparison to the "best company" are:

1. Cooperation with in house developers /guardians is exemplary.
2. In house developers cooperate during the development of the application.
3. Design and contents of documentation that must be handed over is prescribed.
4. Demands are written down (e-mail, paper document).
5. Conditions for work at the buyer's premises are suitable.

One statement were ELES was graded much higher in comparison to the other companies is:

1. Relationship with the to employees at buyer's side is friendly.

Table 5.1: Statements and grades given by the subcontractor employees to ELES in comparison to their best partner in an outsourcing partnership. (In bold type are those statements where the grades differ the most.) - PART I

Statements	ELES's grades	Best other company grades
Requirements of end users are known	3,8	4,44
Requirements of end users are similar to the requirements in other companies	3,6	3,89
Cooperation with end users is exemplary	4,4	4,56
Guardian of the application is known	4,1	4,78
Cooperation with in-house developers/-guardians is exemplary	**3,7**	**4,56**
In-house developers cooperate during development of the application	**3,1**	**4,33**
On the buyers side it is known who decides on quality of the product or application	3,3	3,75
Instructions for producing software are developed	3,1	3,56
Design and contents of documentation that must be handed over is prescribed	**2,6**	**4**
Cooperation is managed by project	3,9	4,22
Project manager on the buyers side is known	4,8	4,77
Project manager at the service provider (outsourced company) is known	4,9	4,89
Responsibilities and competences are clearly defined	3,8	4,22
Processes supported with applications are clearly defined	3,6	3,78
Time limits are clearly defined	4,1	4,11
Time limits are suitable	4,1	3,89
Formality is too large	2,9	2,44
Formality is too small	2,9	2,66
Buyer is satisfied with the product	4,1	4,56
Conditions for work at the buyers premises are suitable	**3,3**	**4,33**
Higher formality of the project is expected	3	2,67
Project manager at the buyers side has enough competences to implement the system	4,5	4,44
Buyer expects, from the supplier "suitable" solutions	4,4	4,56
Handing over the solution is formalized	3,4	4

Table 5.2: Statements and grades given by the subcontractor employees to ELES in comparison to their best partner in an outsourcing partnership. (In bold type are those statements where the grades differ the most.) - PART II

Statements	ELES's grades	Best other company grades
Relationship with the employees is correct	4,7	4,67
Relationship with the employees is friendly	***3,9***	***3,3***
Demands are written down (e-mail, paper document)	**3,7**	**4,56**
Demands and if they can be carried out are discussed together with the buyer	4,4	4,44
Demands are traceable	3,8	4,22
Demands overlap	3,2	3,22
I have a feeling that the buyer is satisfied with our product	3,9	4,56
I use the buyers standards at work	3,4	3,44
I use common standards at work (ISO, BS, IEC,..)	2,5	2,33
I use standards of my own company at work	4,9	4,89

5.3 Discussion

We will discus the six statements more in-depth and we will try to relate these six statements to the relations identified in the organizational structure [46, 43] depicted in Figure 2.5.

1. Cooperation with in house developers - guardians is exemplary. This statement is an example of **Relationship of a Communicational Nature** in Figure 2.5.

It is obvious that besides being friends we need to work together. The foundation for a group to function together is the communication of all team members. Communication is the bearer of all social activities. It enables the individual to use experiences of other members and to learn much easier and faster. Through communication the team analyzes problems, accepts decisions, and adjusts work of individuals to reach the common goal [38].

Team members need to think together to solve the problems up front. People inside the enterprise know the users and the processes, team members from the subcontractor know different ways to solve unpleasant situations and have a broader view on how to solve software problems. If in-house team members do not accept other members as equal and all the time undermine their attempts to change or even to improve processes, it is not stimulat-

ing for the sub-contractor's team members and not good for the developed software. Too much energy is wasted on just trying to get the people to cooperate.

Interactive, face-to-face communication is the cheapest and fastest channel for exchanging information [15].

2. In house developers cooperate during the development of the application. This statement is an example of **Relationship of a Coordinative nature** in Figure 2.5.

Although this statement sounds coordinative, it also depends to a large degree on the in-house people being developers. It is important how employees are prepared to comprehend the technology and methodology and respond to it. The management of external relationships depends on people interacting and how the company looks on such cooperation.

The importance of this relationship is reflected in the faster acceptance of newly developed software into the process, better quality training for the end users, less problems with in-house maintenance (in-house developers know where problems can be expected, they are familiar with the code, there is less time lost when introducing new applications, mistakes can be detected faster, etc.).

3. Design and contents of documentation that must be handed over is prescribed and ***4. Demands are written down (e-mail, paper document).*** This two statements are examples of **Relationship of a Technical Nature** in Figure 2.5.

In engineering a detailed design is a prerequisite for production. For example, without design drawings the production of automotive parts. Without proper design drawings the production cannot begin. Unfortunately, this is not the case in software development! The design of applications can begin even on purely verbal agreements. Programmers, however, usually produce some written material even if only for personal use. This can later bring problems in traceability between demands and solutions.

It is easier and less time demanding if a company prescribes how design, documentation, and context must look like and what they must contain. It's like having a template for writing an article or a book. The team should be given standards to follow, that could be developed in house. This would be the agreement on how to work. The team should know what is and what is not important when producing software, what must be solved on the data base and what in programs, how programs and where programs are executed and who is responsible for what. It is desirable that rules are written down and that new members can read them.

5. Conditions for work at the buyer's premises are suitable. This statement is an example of **Relationship of a Technical/Motivational Nature in Figure 2.5.**

Environment for work is something Herzberg in 1956 classified under the hygienic factor [44]. It must not be absent. The presence of a suitable working environment does not yield satisfaction, but its absence yields dissatisfaction.

Technically and organizationally it is harder to provide a place for work than equipment for work. Today, PCs are not expensive, they can be purchased quickly. The same goes for the operating systems and software used by the developers. A room or an adequate place where developers from the subcontractor company can work when they come to visit the buyer's company is harder to ensure.

6. Relation to employees at buyer's side is friendly. This statement is an example of **Relationship of a Personnel Nature** in Figure 2.5.

This is a purely personal relationship that is established between "our" and "their" employees. It depends mainly on the personal characteristics of the personnel involved on both sides. Such a relationship takes time to develop and to become sincere. A newly chosen subcontracting company cannot expect to be accepted instantly within another established environment. Gradually (over a period of time) it can, under certain conditions, become friendly.

In Figure 5.1 the structure of relationships shown in Figure 2.5 is redrawn, showing just the six relationships that we identified and discussed above. In our survey we identified five statements that make ELES a less desirable partner in the outsourcing part of software development projects. These five statements together with the 6th favorable statement are examples of 5 types of relationships shown in Figure 2.5. To make the organizational structure shown in Figure 2.5 more tuned for achieving quality software solutions, the factors important for IT projects turn the organization towards a more encouraged self-initiative structure. (see Figure 5.2). In ELES they try to put more stress on those factors and they achieve, even with some drawbacks in organization and some other negative influences, as the result of the questionnaire shows, a positive result in projects success.

5.4 Criteria to grade the quality of a project team

In a successful team, team members coming from outside and from in-house must get on well together. We have expanded and build upon the seven (7) criteria of diversity that managers are being force to confront, given in [37]. Using the knowledge of Human Relationship Management [38] and the questionnaire that comes with the MUKOZ [43] methodology, the author of the thesis has identified the following twenty (20) criteria that influence the relationships in a software development project team (not in any particular order):

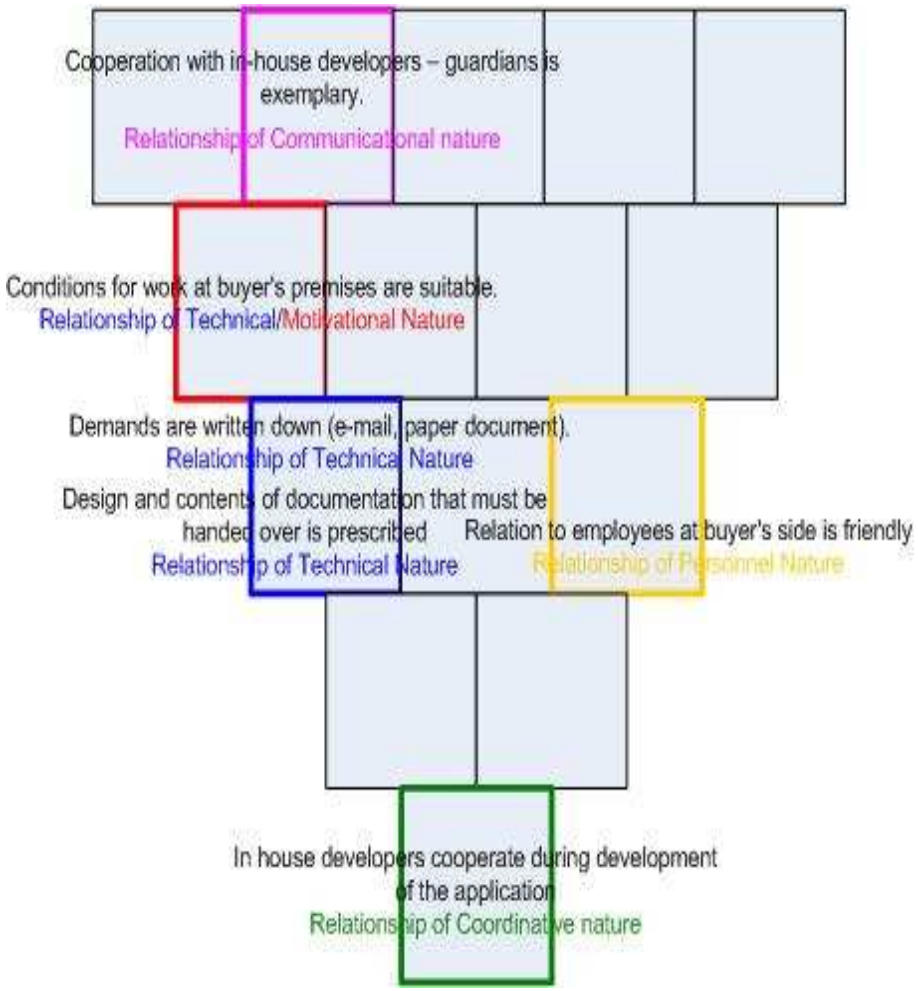

Figure 5.1: Relationship towards subcontracting company which are important for ELES from the structure of relationships shown in Figure 2.5

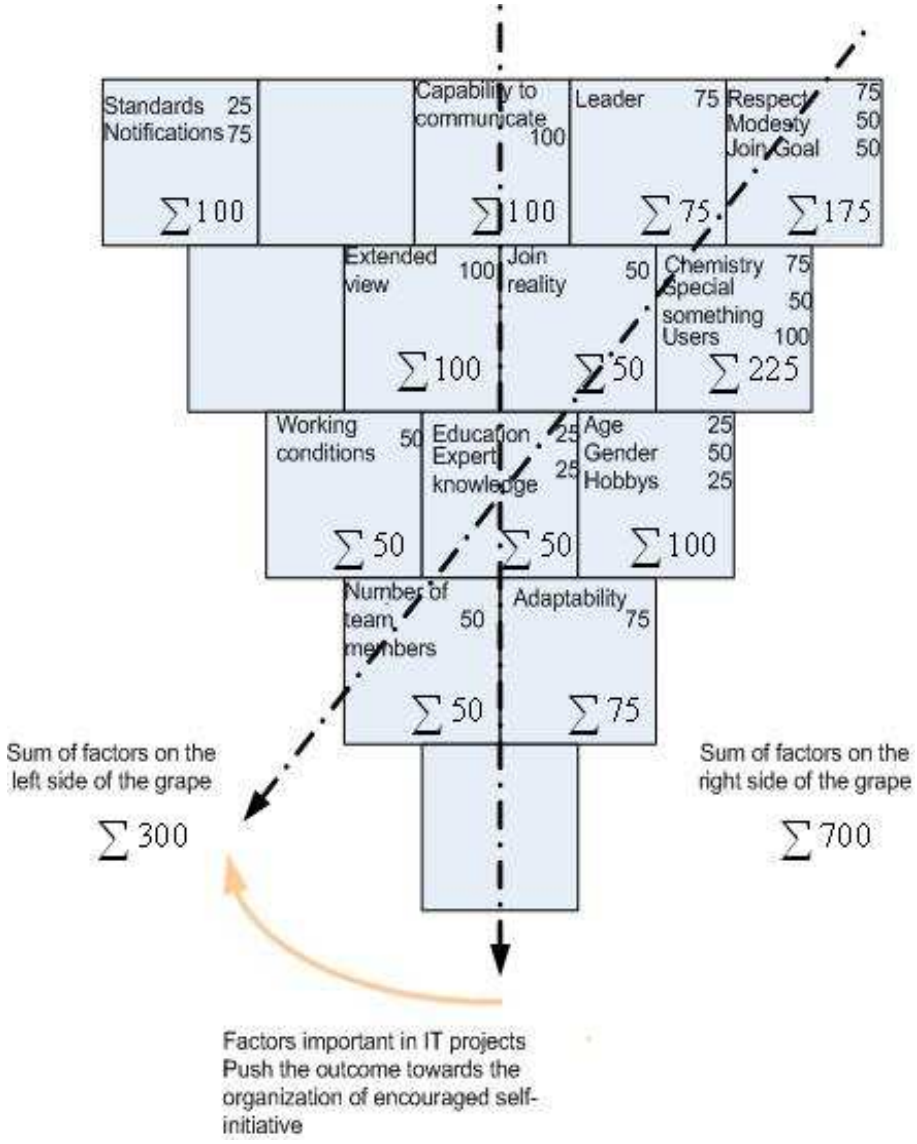

Figure 5.2: Factors important in IT team swing the "grape of relationships" towards a more encouraged self-initiative structure from Figure 2.5.

1. Age

   It is constructive for the team that members are of different ages, and that the project leader is somewhere in the middle. Age gives wisdom and experience that can only be gained in time, youth gives energy and enthusiasm.

2. Gender

   Men and women have different perspectives and a different comprehension of the world. Some tension is like some conflict (different opinions) and can be good for progress.

3. Joint goal

   Synergy of effects can only be achieved if every team member being from inside the enterprise or outsourced, recognize the same goal even if it's seen from a different perspective. In-house employees must give their users working software and so must the sub-contractors employees that form the project team. The final users in a project team strive towards a more informatized every-day job for themselves that will facilitate their working day and give them the information that they need. This communication will only bear fruit if we all know what we are talking about.

4. Capability to communicate

   People on the job must be able to express their goals, wishes, difficulties and ideas. Some people are more extroverted than others and with introverted people, it is difficult (but not impossible) to work. The project leader spends more energy on a person who has difficulties in communicating. To many, such a project team member can exhaust the project manager and other members.

5. Shared hobbies outside the job

   Talking only about work in progress can kill the enthusiasm on the job. Sometimes when talking about something completely different - when the mind wanders, one can find the right idea for the job at hand.

6. Existence of standards

   Today many standards exists prescribing how to act in order to get a quality product. Standards are especially useful when the new team has only just started to communicate. They offer some common ground to work on and provide a basic direction.

7. Working conditions

   When I was faced with the results of the survey in Tables 5.1 and 5.2, I was surprised just how important good working conditions at the

place of the buyer/final user are for the sub-contractor's team members. Even if they only come for several hours each week, putting them in a dark corner does not yield a positive attitude towards the job. Sub-contractor's personnel needs to feel welcome, and one way of showing them that, is to give them a place to work that is comparable with the environment where the in-house employees work.

8. Notifications

   Standardizing notifications or at least the time periods when notifications must circle the team is wise. This is just one method of formal communication which is necessary to keep all involved informed. Even if the team is well integrated, one can forget to tell one person about something he or she has done which could have a critical meaning for someone else. E-mails are a good way to notify others about progress, the next steps, about the reactions of the users, etc.

9. Limited number of team members

   Books often prescribe the ideal number of team members to be from 5 to 12. The actual number depends on the scope of the project, the extent of knowledge of the team members, the capability of the team leader to manage large numbers of people and the way communications have been established. When the number extends beyond a certain limit, the energy needed to communicate is larger then the energy focused to solve the problem. The project leader must be able to notice that and reduce the number in a team if needed.

10. Adaptability

    Gallup Management Journal [78] describes Adaptability as "going with the flow". Adaptable people tend to be "now" people who take things as they come and discover the future one day at a time. This is not necessarily what one expects in a project team. A project manager should make his/her team members their think-tank, they should not just simply follow him or her. Sometimes however, we need to simply adapt, and make the best out of the given situation and discover the future one day at a time. We need to be able to adapt to our co-workers and be ready to accept ideas that are not ours.

11. Extended view

    Having a good project team means also having in the team people that see more than just their own every day work. This is one of the reasons we use outsourcing, not only because we have a shortage in in-house personnel. People from the outside bring a different view to the problems that we encounter. They have a different background and different solutions. The extended view is also something one must

desire to possess. It is best not to be molded into the limits of ones own reality. The best solutions are usually the most simple ones, but we do not see them because they lie in someone else's working area.

12. Chemistry

    Physical, emotional and psychological chemistry in a relationship is very hard to predict. Chemistry is a complex emotion that is hard to explain but easily recognized when it is present. It differs from person to person and is impossible to factor into the process of choosing a project team.

13. Respect

    Respect for the others and for yourself plays a vital role in a team. Some cultures, like the Japanese for instance respect their elders in some cases up to the point where the idea of the freshman does not matter. This is not good for a creative team in software development.

14. Modesty

    Describes a set of culturally determined values that relate to the presentation of one's self to others. One should not go as far as humility.

15. Joint reality

    Joint Reality is inspired by the economic term joint venture. It means to express partnership and respect, the objective overlap of interests and values, the limited collaboration towards common goals, the building of trust based on cooperation and mutual understanding from dialogue in a pragmatic way. But the difference is that Joint Reality is not restricted to the economic target, but it also encompasses all aspects and problems of life.

16. Contact with users

    Users are the ones that will be or will not be satisfied with our final product. The quality of the project team work has no meaning for them and they do not extrapolate the quality of the product from the quality of work. To check if we are on the right track and will develop what they expect, it is prudent to meet with them, to show them our progress and to ask them additional questions. Doing this will in practice constantly change the fundamentals we started working on. This is where we can say we have to work as an adaptive software development team using Adaptive software development methods as described in Section 1.5.1.

17. Leader - project manager

Many books [64, 26, 3, 44, 73, 47, 25] are written on this subject and on the manager's ability to lead. Besides leading, a project manager must also motivate, organize and control. Some skills can be learned and others, one must be born with. A prudent recommendation for project initiators is that the buyer should choose the project manager who can carry out the job.

18. Education

    IT professionals can come from very different educational backgrounds. Education does not give one all the knowledge and wisdom one needs, but it gives the different angle for looking at some things and offers a different approach to problems. In my experience, my team members have been educated in: economics, information technology, sociology, mechanical engineering, electro-technical and telecommunication engineering. It is desirable if a team consists of different profiles of education. This ensure there are different views and brings different solutions. Users come from an even broader list of different educations.

19. Expert knowledge

    When supporting different business processes, a project team needs different experts coming from the process it supports with IT. There is a problem, however, that experts often speak in their own "language" or "jargon", that other team members will have difficulty following. Experts should know what kind of cross checks are required manually to be able to check the results provided by the computer program. They are the ones providing the right formula for the program according to its stated purpose.

20. "That special something"

    This is the most intangible of all factors listed above. We can have all the other facts in perfect combination, but "that special something" is missing. It has to do with chemistry, with gender, education, ... and again it has nothing to do with them. It can even change over time.

We try to get the ideal combination before the real job starts. Usually with the scale of projects described in this work, there is no time and money to perform some psychology testing on future members of the team. One of the guidelines can be, if we worked well together once, there is a strong possibility we can do it again. On the other hand, some people do not fit well together not even on the first instance.

The project leader must be given the possibility and should have the competence to chose the team members and only then he or she can be responsible for how the team acts and for the outputs.

Based on the results of the field survey and considering the best practices in software development [15, 53, 66, 87, 10, 21, 4, 52, 1, 47, 35], the diagram showing influencing factors on the project as presented in Figure 5.3. The factors shown were weighed by the author of the thesis (based on the influence of relationships identified in the field survey (Tables 5.1 and 5.2)) and then included into the model of relationships (Figures 2.5 and 5.1) to finally draw the ideal relationships in a software development team as depicted in Figure 5.2.

Some of the twenty factors are more important for achieving quality then the rest. We have divided the twenty factors into the following four subgroups.

1. Biologically defined (red color in Figure 5.3) are Age, Gender, Adaptability , Chemistry and Special Something. Those factors are something we bring into this world when we are born and do usually not changes through our entire life. When we get older we get wiser, and we may become even less adaptable, while chemistry is changing as we change. Maybe we work easier with people our own age? These are tangible factors in their essence and are very important.

2. Capabilities (orange color in Figure 5.3) are Capability to Communicate, Extended View, Respect, Modesty, Leader and Hobby. We learn these, not necessarily in school but through experience and by making mistakes. Some people are born leaders but they must be given a chance to lead a team that will embrace his or her leadership.

3. Trained (blue color in Figure 5.3) are Education, Expert Knowledge, Joint Reality, Joint Goal. We go to school and get a formal education. This is the easiest to check. These factors are not as important as the ones gained through experience.

4. Outside influences (green color in Figure 5.3) are Standards, Working Conditions, Number of Team Members, Users, Notifications. Hopefully one can provide suitable conditions. This factors can be counted, predicted, found and established. These are the most suitable for formalization and standardization of project work.

5.5 Adaptive software development as the proper methodology

The characteristics of Adaptive Software Development (ASD) process are:

1. mission focused,

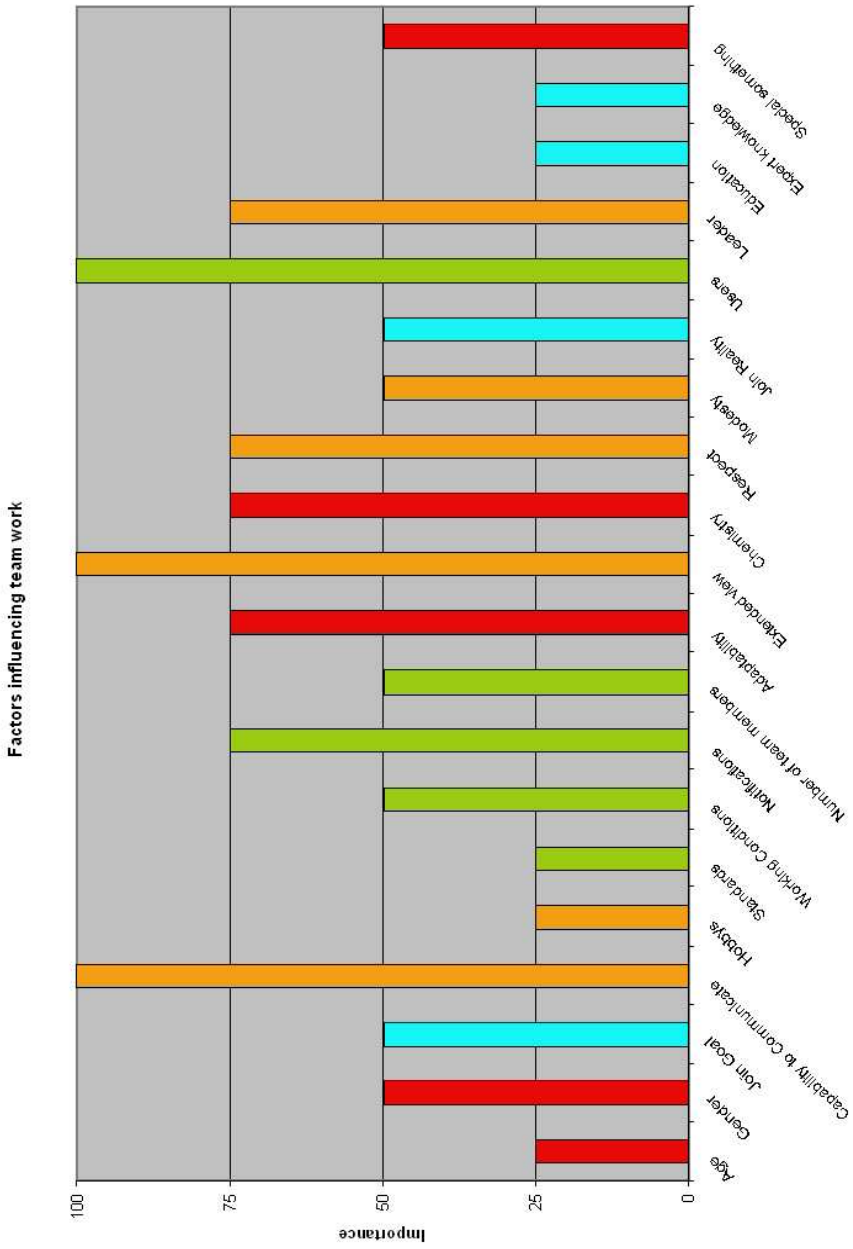

Figure 5.3: Factors influencing relationships in a project team

2. feature based,
3. iterative,
4. time boxed,
5. risk driven, and
6. change tolerant.

When a company like ELES sub-contracts a company to cooperate in a software development team, this project is mission focused. Technical specification of the project that must be presented in the public procurement process assures that. The same goes for the time limits that are actually set, because subcontractor competes with how fast one can bring the project to an end (if they offer the same price).

In Section 1.5.1 we suggested that for the process of software development in a company together with a subcontractor Adaptive Software Development or ASD for short is the most suitable. It was developed by James A. Heighsmith III and published in year 2000 [23]. ADS replaces the traditional waterfall cycle (see Figure 2.4) with a repeating series of Speculate, Collaborate and Learn cycles. Speculation comes instead of Planning. In speculation uncertainty is no longer a weakness. Collaboration highlights the importance of relationships and communication within a project team. Learning stresses the need to acknowledge and react to mistakes [1]. In our every day job, we have to focus more on the results and their quality, than on the task and the processes used for producing the result. In a team, we constantly have to meet the final users, give instructions to the in-house and outsourced developers so we have to reach one level of understanding and learning from each other. Here is where the good relationships come in to play the vital role. People, relationships and uncertainty are the central constituents of adaptive software development. This method has no built-in limitations for its application. In practice, it leaves many details open. But this in practice is how our case really is.

Highsmith [23] states some important facts about ASD and the core statement being put in front of software development is: In building business software, requirement changes are the norm, the question is what we do about it. Hightsmith's answer is **"Put the people first".** My contribution to that is "**Put the *relationships between* the people first**". Fowler [19] states that in an adaptive process the customer - buyer has tight control over the software development process. At every iteration, they get both to check the progress and to alter the direction of the software development. This leads to a much closer relationship with the software developers, a true business partnership. When there is a third partner - a subcontractor, the relationship must be close with them too. Execution requires a very effective

team of developers. The team needs to be effective both in the quality of the individuals, and in the way the team blend together.

Neither the people involved nor the relationships they form amongst themselves can be considered replaceable parts or simply a resource (Frederick Taylor's Scientific Management approach [44]) for the project. A project is built from people having different skills, working in a physical environment within an organizational culture [14]. People are not predictable but they are the most important factor in the software development. Developers (in-house and outsourced) must be able, during their interactions, to make all technical decisions. Both need very close contact with business expertise that is in our case seen as a third party. Since developers are capable professionals in their own discipline, they need to be able to work as equals with other professionals in other disciplines. The relationship team members are able to establish, again play the vital role.

Factor that we identified in Section 5.4 enable that relationships between team members can cope with the risk (adaptability, joint goal, expert knowledge) and changes (notifications, extended view, etc.). The trust must be established and joint decisions must be made. Trust makes it possible that developers choose the adaptive way of working.

In the unpredictable environment within which we work during the course of our every-day business, we need people to collaborate in a special way in order to deal with the uncertainty. Management attention is less about telling people what to do, and more about encouraging communication so that people can come up with creative answers themselves. It is not about giving the project team all the answers but it is about encouraging them to ask the right questions. This is the main problem which is also stated in the world famous book Hitchhiker's Guide to the Galaxy by Douglas Adams [2]. Part of the art of project management is learning how and when to trust people and when not to trust them. Part of the art of methodology design is to learn when formal constraints add more burden than benefit [15].

Chapter 6

Conclusions and contributions of the thesis

The research environment for this thesis is the Slovenian utility enterprise Elektro - Slovenija d.o.o. (ELES). ELES is organized in sectors and one of them is the Sector for Business informatics. The Sector for Business informatics has 18 staff members, 8 of them are software developers and analytics that take care for the ELES business information system. This number of software developing personnel is not high enough to satisfy the needs for developing new applications or for vital changes in existing applications. In such cases ELES has to find a subcontractor who does part of the software development project. Team members from the subcontractor that participate in such projects are predominantly programmers only. The analysis of the problem and specification of the solution are done in majority by in-house staff.

The enterprise is publicly owned and as such subjected to the laws on public procurement. This means that the subcontractor and the additional members for the software development team must be chosen in accordance to the process specified in the Public Procurement Act [94]. The problem that arises from that fact is, that the buyer (ELES, Business Informatics department) can not choose the subcontractor by simply choosing the preferred partner who has the competences to perform the job. The criteria for choosing the subcontractor must be objective and formal. The hypotheses of this thesis presented in Section 1.6 should help to answer the dilemma how one can get the most out of the situation even in these constricting conditions.

The results presented in Chapter 5 might lead the way to a more process and product oriented public procurement. Hypotheses also state that there are more important issues in software development team than technical specification and lowest price. These issues are people in the project team and relationships they establish during the cooperation.

6.1 Recapitulation of the thesis

To be able to confirm or deny our hypotheses we concentrated on three different but in practice overlapping terms: Quality (described in Chapter 2), Project Management (described in Chapter 3) and Outsourcing (described in Chapter 4). Each Chapter tries to focus on the possible solution for the stated problems - relationships between project team members.

An old team can become weary, new teams are unpredictable. The quality of a program is, as mentioned through this entire thesis, something very undefined. I give some general orientation as to what to look for when forming a team to develop software. If you get the chance to choose the team members or when they are assigned to the team, what to look for in them and to stress their advantages.

In Chapter 2, Quality is introduced. As proven later not just the quality of a product or service is important. The quality of the organization as a joint set of relationships is also an important factor for the overall quality of work. Standards are supposed to facilitate the quality but occasionally they sometimes represent more of an obstacle. Models such as the Waterfall Model, Quality function deployment, UML modeling, ITIL with Service Level Agreement, etc., are used in large software development companies where developing software is the core business of the enterprise.

Small IT departments in enterprises whose core business is far from software development who use information technology for supporting main processes, the situation is noticeably different. The IT staff are pressed from one side by the internal customers and internal final users and then pressed on the other side by the sub-contractors. Teams are smaller and therefore the relationships between team members that usually come from inside the company and from the subcontracting enterprise become even more important. The quality of those relationships is connected to the quality of work and as such to the quality of the final product or working application. In this thesis some guidelines are proposed to choose the right team to prepare the right conditions for a good relationship.

Relationships most influencing the quality of software are:

1. Relationship of a Communicational Nature,
2. Relationship of a Coordinative Nature,
3. Relationship of a Technical Nature,
4. Relationship of a Technical/Motivational Nature,
5. Relationship of a Personnel Nature.

Project management is described in Chapter 3. Most software development is done in form of projects. Projects are defined by time, resources and

costs. The fourth dimension is quality. Project management introduces the terms project team and team members and gives them responsibilities and competences. Models for calculating the cost of a large software development project such as COCOMO do not include quality in their calculations.

Differences in software development projects and other engineering projects such as construction and mechanical engineering are discussed as they are the important factors in different approaches towards team members and their relationships. The different phases of a project are important if we are discussing a software development project or some other type of project. Predictability is almost impossible to target when considering software development.

Chapter 4 is dedicated to outsourcing. Because everyone tries to do what he knows best, many enterprises outsource some part of the necessary businesses so that outsourcing IT is not something unknown or new. Although IT can be outsourced completely, and only data input remains at the enterprise, we didn't look into such cases in this thesis.

IT departments like the one described in this thesis are not rare in Slovenian enterprises. The enterprise has typically a few analysts and some programmers that develop and maintain applications. This organization is necessary to fulfill the smaller demands of the customers immediately. Analysts are the ones that supposedly know the process best and have the knowledge to know how to support them with software. Naturally, development goes forward and a few times a year, larger software development projects are necessary. In the case of publicly owned enterprise, some procedures are obligatory by law and must be followed to be able to choose the subcontracting company that will support us in the development.

A survey of sub-contracted personnel who worked on different projects was carried out and is described and analyzed in Chapter 5 . Statements were graded to identify which are the issues that should be stressed in a combined project team for software development and to prove our hypotheses. The sub-contracted personnel were asked to grade statements about how they see the organization that they worked for as sub-contractors, what they like and what they dislike.

On the basis of their answers with regard to documented studies and experience, we created a model of important characteristics of project team personnel that comprises 20 factors organized into four categories that describe the personality of every team member and that, in author's opinion, influences the quality of project team relationships. Next, factors were weighted by their influence on relationships and entered into the Methodology of establishing the quality of firms' organization. The quality of relationships indirectly influences the quality of the finished application [43]. When perfect conditions are given, and the team has all desirable characteristics the organization becomes an encouraged self-initiative organization (see Figure 5.2). The importance of the factors on the right side of "the grape" turn

it towards the pleasant organization (compare with Figure 2.5) [28, 44, 46] . In such an environment better products can be developed.

In Chapter 5 we propose the Adaptive Software Development (ASD) as the best method of developing software in this environment.

6.2 Hypotheses revisited

Hypothesis No. 1: *Relationships formed in a project team before and during development of an information system or application are important factors influencing the quality of the final product, but hard to plan and predict in advance.*

Relationships formed before, during and after the software development projects are important. They are something that cannot be predefined and standardized. But they can be predicted, reused and improved upon. Development of software applications for the known final user has its basis in formal procedures and processes. Knowledge that comes into the project with the project team members is mainly gained on a formal level. How it is used is defined differently in each case.

We used the Methodology of assuming the quality of organization - (MUKOZ) [43] to bind the answers we got in the field survey at ELES' subcontracting companies in 2003, with the importance of relationships between team members on software development project. The result proves that soft factors of the project turn the relationships grape towards the friendly organization or organization of encouraged self-initiative. Relationships are formed through cooperation on the project and outside it. Unfortunately due to the fact that soft factors (human side of project) is more important on projects discussed in the text then the technical side, those factors are hard to predict and subjected to many influences. Additionally, due the limitations of Public procurement act factors influencing the quality of relationships can not be stressed in the process of choosing the subcontractor to whom the work is outsourced.

Human Resource Management (HRM) at buyers and at sub-contractor's side, plays a much bigger role in software development that one would imagine. The project team leader must or should have the privilege to choose his or her own team. The project leader should be able to combine not just the right people for the right job, but to combine team members to get the best synergy. Authors like Instead, Fulop and Liley [37] not only highlight the differences but celebrate them too. Differences are based on age, race, gender, sexuality, ethnicity, beliefs, experience, disability and so on, although often gender and race receive the most attention. These differences have to be accommodated or even celebrated in management. The power of a team becomes embedded in relationships - the mortar between the stones in a wall. Figure 6.1 represents, with rocks forming a wall, the difference

between a homogenous and a heterogenous group in a project team. People are different. It is not productive to look for an ideal person to fit the wall exactly like the precisely carved stones on the left side of Figure 6.1. Instead, relationship between team members are the bond that ties the wall together. Key themes for the analysis become the project culture and its relationships [13]. Managers have to deal with multiple realities, roles and identities, and the multiple loyalties of individuals. In the case of a project, the project manager is the one that should bring such relationships into life that are of mutual advantage. Without building, maintaining and developing relationships, that are constantly changing, a manager cannot manage.

Stones are precarved to fit perfectly together.
Replacement is hard to find.
Homogenous structure.

Stones are as they come. Gaps are field with relationships.
Heterogeneous structure.

Figure 6.1: Rocks representing different types of teams.

The hypothesis is confirmed although more analysis of different projects in different enterprises should be carried out in the future.

Hypothesis No. 2: *The combination of ideal personal and professional characteristics of project team members yields better products.*

A model of important personal and professional characteristics of team members is developed. When the results are weighted and put into the proposed methodology [43] the ideal structure of factors characteristic of a person is visible. Factors are classified into four categories. Because there is no ideal person, the model stresses that the project leader should balance between the characteristics of his team members to develop the positive and not to entice the negative. The model should serve as an instrument for the project manager or better yet, for the project owner to find the right people for the roles they perform on the project or at least encourage the

development of the relationships that will contribute to the quality of project results. (See hypothesis No. 1).

When we do not have the possibility to choose the project team without outside influences, we cannot beforehand ensure that our project team will not develop a synergy of positive effects. Some of the factors can not be chosen (users) but we have to allow for them anyway. The project is not necessarily damned from the start in advance, but when the team is predetermined, at least we know where we can find the obstacles and try to prevent misunderstandings.

The hypothesis is confirmed although there are many other influences on the final product of the software development team.

Hypothesis No. 3: *Adaptive software development method is highly suitable for the environment which is the object of study in this dissertation.*

Adaptive software development (ASD) as one of the method in agile software development methodology is due to its characteristics and due to its emphasis on the human side of the software development project one of the best suitable methodology that Slovenian enterprises can use. ASD is also change tolerant and iterative. Large, highly formal projects can hardly cope with fast changes and much iteration. The project team must be flexible enough, have enough coordination and cooperation to manage and perform without a lot of formality. When one cannot do it alone, it needs to use the relationship established towards the others.

Being aware that the method we are using in every day software development is really very similar to the adaptive software development method, changes the formality of our work. Uncertainty is no longer our enemy and drawback, but instead we tend to use it in a positive way. If inputs to the project are uncertain, so can the process be. We can and must change the way of thinking and programming. This, however, is not what standards want us to do in the first place. This is not something even our supplier that will become our subcontractor would expect from us. How can we put in the tender that what we are looking for, is not the product or the service but the people that form the team that makes the product through the service of programming?

The hypothesis cannot be denied. Probably there is a different ideal method to develop software for every enterprise environment but ASD is suitable for the environment found at ELES.

This thesis puts forward the relationship between people in a software development project team. It considers standards, theory and other best practices (which are described in the work). It is not often asked how people react, what triggers them and how this influences the quality of their work and the quality of the software at the end. In this thesis I tried to answer this questions.

Dodatek A

Povzetek v slovenščini

V slovenskih podjetjih, ki imajo oddelek za informatiko, izdelamo veliko uporabniških rešitev, potrebnih za podporo poslovnim procesom, in to v projektnih skupinah, sestavljenih iz "hišnih" analitikov in programerjev ter izvajalcev nalog na projektu iz zunanjih podjetij. Razmerja med člani projektne skupine, ki jih sodelujoči vzpostavimo med delom ali izven njega, pomembno vplivajo na kakovost samega izdelka uporabniške rešitve.

A.1 Trenuten položaj razvoja uporabniških rešitev

Vsaka združba (podjetje) v sodobnem poslovanju vključi v svoje poslovanje vsaj nekaj storitev, pomembnih za uresničitev svojih ciljev, drugih združb. To velja tudi za storitve, potrebne v informatiki. Pri tem išče odgovore na vprašanja: Kako zadovoljiti potrebe uporabnikov? Kako ustreči naročnikom uporabniških rešitev, s katerimi želijo pospešiti delo ali prihraniti kakšen evro? Kako želje sodelujočih v poslovanju natančno pretvoriti v programsko kodo? Kako izbrati zunanjega dobavitelja, ki nas bo stal čim manj, zagotovil pa nam bo kakovostno rešitev oziroma pripomogel k njej?

Predmet raziskave teze je slovensko javno gospodarsko podjetje Elektro-Slovenija d.o.o. (v nadaljnjem besedilu ELES). V slovenskem prostoru podjetje skrbi za prenos elektrike na visokih napetostih in za stabilnost slovenskega elektro-energetskega sistema. V podjetju je 500 zaposlencev, od tega jih je 18 zaposlenih v Sektorju poslovne informatike, ki je bil za avtorico ožje področje raziskave.

V slovenskih podjetjih naletimo na več načinov obvladovanja in razvoja uporabniških rešitev, namenjenih podpori poslovnim funkcijam in procesom. Velikost Sektorja za poslovno informatiko je običajno odvisna od tega, ali podjetje kupi rešitev, kot npr. Oracle e-Business Suit, SAP, Navision ali pa, bodisi v celoti, bodisi delno programira rešitve z različnimi orodji in na različnih bazah podatkov. V obeh primerih se pojavlja oddajanje izvajanja določenih nalog zunanjim izvajalcem (angl. *outsourcing*), saj podjetja, ki

jim programiranje ne predstavlja temeljne dejavnosti, ne razpolagajo z vsemi znanji in dovolj velikim številom ljudi, potrebnih za izvedbo večjih razvojnih projektov.

Obvladovanje projektov (angl. *project management*) je pravzaprav oblika poslovnih procesov, ki je poznana predvsem v gradbeništvu in povezanih dejavnostih, ko govorimo o gradnji cest, zgradb, izdelavi turbin za elektrarne,... Na področju informatike je obvladovanje projekta razvoja uporabniške rešitve precej abstrakten pojem. Dejstvo je, da pri inženirskem projektu sama priprava projekta stane mnogo manj in traja krajši čas kot njegova izvedba. V informacijskih projektih je pripravljalni čas neredko daljši in zahteva vključitev visoko izobraženih zaposlencev kot prvin poslovnega procesa. Izdelek kot uporabniška rešitev je za večino nekaj neoprijemljivega, kakovost tega izdelka pa težko določljiva. Uporabniška rešitev ima namreč nekaj določil storitve, kot so zanesljivost, varnost, dostopnost in razumevanje strank, kar niso vedno običajne lastnosti izdelka. Če se na inženirskih projektih pogovarjajo gradbeniki in nosilci sorodnih poklicev med seboj, se morajo na projektih informacijske tehnologije pogovarjati ljudje različnih strok z raznolikimi specialnimi znanji. Pogosto to predstavlja večji problem v sporazumevanju kot pogovor dveh ljudi z različnim maternim jezikom.

Naloga, ki sem si jo zadala, je bila razviti način ali metodo, opredeliti osnovne dejavnike, ki lahko pripomorejo k uspešni "mešani" projektni skupini, ki razvija poslovno uporabniško rešitev.

A.1.1 Javna podjetja in javno naročilo

Podjetja v javni lasti pri naročanju zunanjih storitev omejuje Zakon o javnih naročilih (Ur.l. RS št. 39/2000) [94]. Zakon predvidi pogoje, ki jih predpiše naročnik in jih morajo izpolniti dobavitelji. Najbolj izpostavljeni pogoj je vedno cena in pa bolj ali manj določene tehnične zahteve za projekt. Avtorji zakona so se pri zahtevah nedvomno posvetili "običajnim projektom", povezanim predvsem z naročanjem opreme, manj pa so imeli v mislih projekte razvoja uporabniških rešitev.

Opredeliti zahteve uporabnikov v jeziku razvijalcev je v okvirih, ki jih postavlja zakon, dokaj težko. Vsi, ki so kdaj pisali take zahteve, vedo, kako zelo se tekom projekta zahteve spreminjajo in kako težko je naknadno prilagajati uporabniške rešitve novim pobudam končnih uporabnikov. V tezi prikazujem drugačen pogled na javne razpise. Zakonsko namreč ni dopustno, da diskriminatorno predpišemo, s kaknimi ljudmi želimo delati. Ob tem, ko predpišemo zahteve glede strokovnih znanj zunanjih izvajalcev, ni v navadi zahtevati od njih določenih značajskih lastnosti, ki pripomorejo k uspešnosti projektne skupine. Zavedati se moramo, da zgolj predpisati, katere standarde naj pri delu zunanji sodelavci uporabljajo, nikakor ne zadošča. Tako je cena običajno tista, ki odločilno vpliva na določitev zunanjega dobavitelja, v našem primeru izvajalca storitev razvoja (analize in programiranja) uporab-

niške rešitve. Na Sliki A.1 je predstavljen miselni vzorec, povezava med dejavniki, ki vplivajo na kakovost projekta.

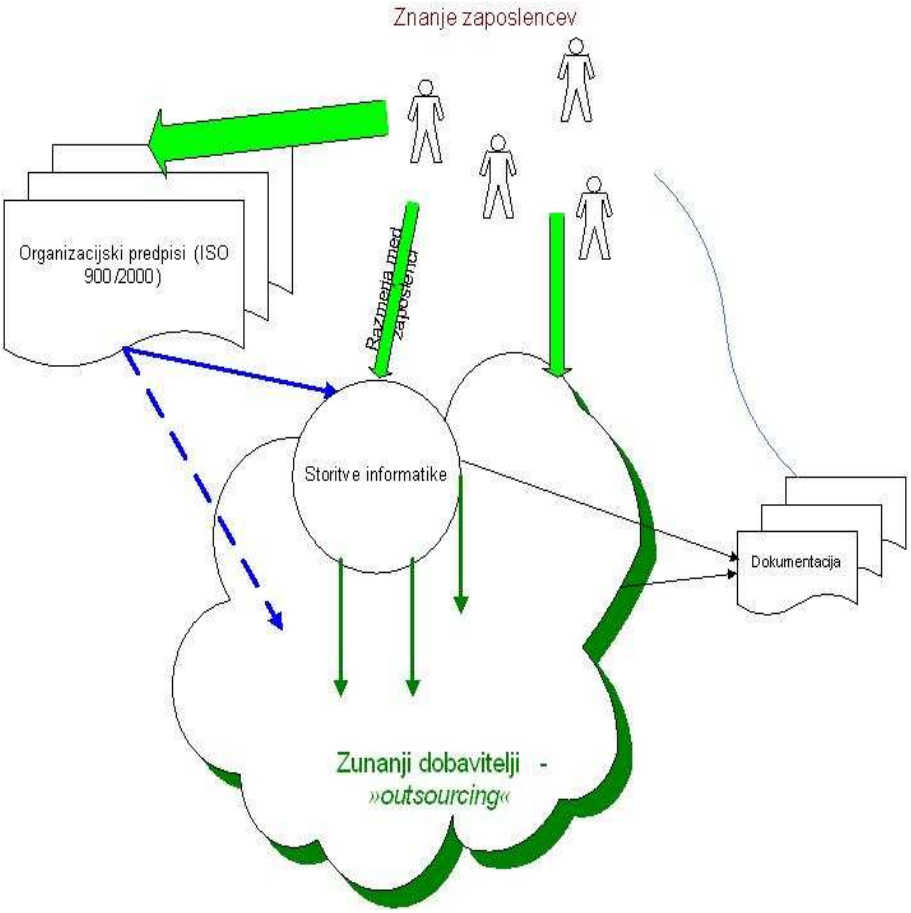

Figure A.1: Prikaz vplivov na kakovost projekta

A.1.2 Hipoteze

Izkušnje iz dela na projektih v podjetju ELES vodijo do naslednjih treh hipotez, ki sem jih razdelala v tezi in jih skozi teorijo in praktično raziskavo tudi podprla:

1. Razmerja, ki nastanejo v projektni ekipi za razvoj programske opreme pred in med delom, so pomembni dejavniki, ki vplivajo na kakovost končnega izdelka, jih je pa težko načrtovati in jih v naprej predvideti.

2. Kombinacija idealnih osebnih in strokovnih značilnosti članov projektne ekipe pripomore h kakovostnejšemu izdelku.

3. Prilagodljiva metoda razvoja programske opreme (ang. *Adaptive Software Development*) je zelo primerna za dano poslovno okolje.

A.2 Standardi in metode dela

Informatiki skušamo s sodobnimi orodji podpreti poslovne funkcije in procese. Ti so običajno popisani v priročniku kakovosti, izdelanem po standardu ISO 9000/2000 [89]. Vendar pa opredeljenost procesov znotraj standarda ni dovolj podrobna, niso popisane razne izjeme in običajno niso predpisani učinki posameznega procesa, najsi gre za poročila o davku na dodano vrednost, ki jih oddajamo Davčni upravi RS, ali plačilni list zaposlenca. ISO standard torej razvijalcem uporabniške rešitve ne zadošča, poda jim le osnovno usmeritev procesa.

Razvoja uporabniških rešitev se tako ali drugače dotika še precej mednarodnih standardov, ki pa v okoljih razvoja uporabniških rešitev v slovenskih podjetjih niti ne predstavljajo temelja ali opore pri razvoju niti ne dajejo smernic za razvoj.

V slovenskem okolju postajajo priljubljene smernice, opredeljene v ITIL (Information Technology Infrastructure Library) [87]. Ta dokument se ne ukvarja toliko s procesi, ki jih informacijska tehnologija podpira, pač pa z organiziranjem same dejavnosti temeljnih procesov v oddelku informatike. ITIL se opira na besedo "sledljivost". Gre za sledljivost sprememb v informacijski infrastrukturi in uporabniških rešitvah. ITIL se ne ukvarja s sestavo projektnih skupin, primernih razvoju, niti ne podaja kakršnihkoli smernic v ta namen. Za premostitev problema pomanjkanja ustreznega standarda je v novembru 2005 izšla serija standardov ISO/IEC 20000 *Service Management Standard*s [85].

Jezik, ki ga teorija pogosto predvidi in naj bi olajšal sporazumevanje med uporabniki in razvijalci uporabniških rešitev, je UML (Unified Modelling Language) [18]. Uporablja se v objektnem programiranju, saj naj bi zagotavljal uporabo najboljše prakse iz svetovnih razvojnih hiš. Vsakdanjost je nekoliko drugačna. Slovenska podjetja namreč uporabljajo relacijske baze in pogosto nimajo strokovnjakov, ki bi se znali sporazumevati v UML-u. Ko poteka razvoj nenadzorovano ali ad-hoc, se nihče ne ukvarja z risanjem UML diagramov. Kadar pa je zakon sprejet tako rekoč za nazaj, ga je treba informacijsko nemudoma podpreti in vsak se znajde po svoje.

EMRIS (Enotna metodologija razvoja informacijskih sistemov) [35] je v slovenski državni upravi poznana, vendar jo v prakso podjetja, razen teorije same, niso prenesla v večjem obsegu. Informatiki jo ocenjujejo za nekoliko "postopkarsko" in zamudno v primerih popravkov ter potrebe po razvoju, kadar je treba tega zaradi zunanjih dejavnikov udejaniti dobesedno preko noči.

A.3 Agilna metodologija razvoja

Leta 2000 je Alistair Cockburn s soavtorji[1], razvil in objavil metodologijo imenovano "Agile Software Development" (v slovenskem prevodu: *"Prilagodljiv razvoj programske opreme"*) [15, 14, 19, 1]. Metodologija naj bi ustrezala sodobnim zahtevam poslovanja v nenehno spreminjajočem se, negotovem in nepredvidljivem okolju. Sporočilo metode, ki so ga podpisali avtorji, pogojujejo s štirimi predpostavkami:

1. Posamezniki in sodelovanje imajo prednost pred procesi in orodji.
2. Delujoči programi imajo prednost pred izčrpno dokumentacijo.
3. Sodelovanje z uporabnikom ima prednost pred pogajanji o pogodbi.
4. Hiter odziv na spremembe ima prednost pred sledenjem načrtom.

Ljudje postanejo pomembnejši od predpisov, spodbujeno je sodelovanje med ljudmi, ki sodelujejo pri razvoju nove programske opreme. Motivacija posameznika in skupine je nekaj, kar je potrebno spodbujati, ne pa omejevati s tehničnimi podrobnostmi. Metoda gotovo moti vse, ki želijo postopke predvsem dokumentirati in prisegajo na neprekinjeno sledljivost. Ti bi se morali zavedati, da v tem primeru govorimo o programiranju in ne o rezultatih nekih meritev.

Ena od možnih metod, ki jih predstavi Agilna metodologija razvoja je t.i. metoda "Adaptive Software Development" ali v prevodu prilagodljiva metoda razvoja programske opreme [23]. Metoda vsebuje nekaj idealnih značilnosti, ki jih podjetja v javni lasti ob naročilu izpolnjujejo že sama po sebi; na primer, naloga je usmerjena, naloga je časovno omejena, ljudem na projektu so dodeljene vloge. Veliko vlogo v omenjeni metodi imajo ljudje, kar pa teza tudi dokazuje.

Vsaka formalna metoda razvoja naj bi v podjetju prispevala k večji učinkovitosti razvoja uporabniške rešitve in njeni kakovosti [66]. Večina podjetji pa metode ne formalizira in deluje v skladu s svojo "dobro prakso", ki se seveda razlikuje od podjetja do podjetja tako, kot se razlikujejo ljudje v skupinah in njihova zgodovina sodelovanja pri razvijanju uporabniških rešitev.

A.4 Razmerja in teorija organizacije

Po Lipovcu [39] predstavlja organizacija sestavo razmerij. Mihelčič [46] predvidi, da vsako podjetje nastane zaradi razmerij tehnične narave, saj

[1] Ostali avtorji metodologije in podpisniki manifesta so: Kent Beck, Mike Beedle, Arie van Bennekum, James A. Highsmith, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland in Dave Thomas.

ni drugega pomembnejšega razloga za ustanovitev podjetja, kot je možnost ustvarjanja in prodaje proizvodov ter storitev. Rosabeth Moss Kanter [47] pravi, da so, podobno kot razmerja med ljudmi, tudi poslovna družabništva živi sistemi, ki imajo nešteto možnosti. Združbe, ki bodo znale izrabiti te možnosti ter učinkovito obvladovati svoje povezave, bodo okrepile svojo vrednost v sredstvih/naložbah. V Sliki A.2 je predstavljen grozd organizacijskih razmerij. Grozd je rezultat študije, izvedene na Gospodarski Zbornici Slovenije leta 1988. Študija je podana v [43]. Prikazan je preplet petih vrst razmerij (tehnične narave, kadrovske narave, koordinacijske narave, komunikacijske narave in motivacijske narave) ter združitev po dveh izmed njih, ki jih zasledimo v vsaki združbi in tudi v projektni skupini. Glede na zaznane poudarke na posameznih razmerjih ali kombinaciji razmerij vemo, da se organizacija združbe ali skupine nagiba bodisi k poslovnim učinkom (proizvodom oziroma storitvam) bodisi k ljudem in delu z njimi.

Govorimo o razmerjih v skupinah oddelkov informacijske tehnologije znotraj podjetij, katerih osnovna dejavnost ni povezana z razvojem programske opreme. Na zaposlence v takem oddelku pritiskajo na eni strani notranji odjemalci (zaposlenci v drugih oddelkih, ki informacijsko tehnologijo uporabljajo za učinkovitejše opravljanje svojega dela), notranji naročniki (vodje oddelkov in direktorji, ki, če ne drugega, potrjujejo letne gospodarske načrte in s tem odobrijo sredstva za delo oddelka informacijske tehnologije) ter zunanji dobavitelji, ki se borijo za svoj zaslužek in sestavljajo pri razvoju uporabniških rešitev pomemben del razvojne skupine. Projektne skupine so majhne, zato so razmerja, ki jih vzpostavijo člani, toliko pomembnejša. Kakovost medsebojnih razmerij, ki se kažejo samostojno oziroma enovito na enega od navedenih načinov ali kot kombinacija njih dveh, pomembno vpliva na kakovost rezultata projektne skupine. V tezi je predstavljena idealna kombinacija razmerij, ki jih potrebuje skupina za razvoj programske opreme.

A.4.1 Vplivni dejavniki

Pogled razvojnika je usmerjen h kupcu. Manj pa je pogled kupca/uporabnika usmerjen nazaj k dobavitelju. V mešanih razvojnih skupinah se morajo vsi člani počutiti dobro in težiti k skupnemu cilju. Uporabnik ni samo nekdo, ki plača račun za opravljen razvoj. V primeru razvoja s pomočjo zunanjega izvajanja dejavnosti namreč nastopata dve vrsti "kupcev"; programer in končni uporabnik iz podjetja naročnika.

V letu 2003 smo izvedli anketo med dobavitelji izdelave (programiranja) uporabniških rešitev za podjetje v javni lasti. Dobavitelje smo prosili, da primerjajo ELES z najboljšim podjetjem, ki mu dobavljajo enako storitev. Ugotovitve so pokazale, da dobavitelji, ki pogosto opravljajo svoje delo v prostorih naročnika in se srečujejo s programerji v podjetju ter uporabniki

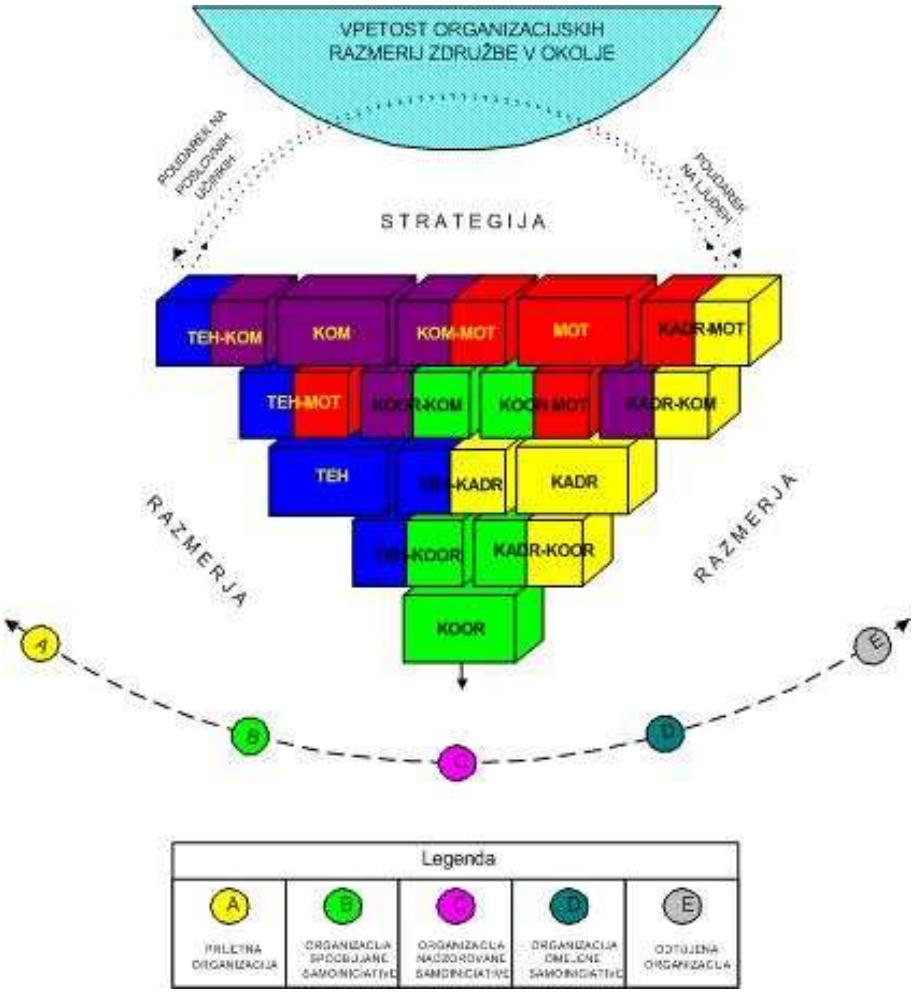

Figure A.2: Grozd organizacijskih razmerij

razvitih uporabnšikih rešitev, posvečajo največjo pozornost naslednjemu:

1. Sodelovanju s "hišnimi" programerji in skrbniki uporabniških rešitev;
2. Ob samem razvoju hišni programerji aktivno sodelujejo in sproti prevzemajo (dele) rešitev;
3. Predpisano obliko predane dokumentacije ob zaključku projekta;
4. Ugodne delovne pogoje (prostor, svetloba, delovni pripomočki,);
5. Zapisane zahteve.

Kljub "pripombam" pridobljenim skozi anketo, so projekti uspešno izpeljani. Ko potrebe zunanjih dobaviteljev uvrstimo v prikazani organizacijski

grozd razmerij, ugotovimo, da je poudarek na dobrem počutju in s tem na občutku smiselnosti v delo vložene večje energije v razmerjih komunikacijske in motivacijske narave. Iz teorije na podlagi odgovorov ankete izvedemo štiri vrste dejavnikov, ki vplivajo na kakovost razmerij v projektni skupini (glej tabelo A.1). Številke v stolpcih pomenijo težo posameznega dejavnika.

Tabela A.1: Vplivi in njihova teža na kakovost razmerij v projektni skupini
Opomba: Številke v stolpcih "vpliv" so določene na podlagi pomena posameznih dejavnikov na kakovost razmerij v projektni ekipi.

Biološko opredeljivo	***Vpliv***	***Sposobnosti***	***Vpliv***
Starost	*25*	*Komuniciranje*	*100*
Spol	*50*	*Širok pogled*	*100*
Prilagodljivost	*75*	*Spoštovanje*	*75*
Kemija	*75*	*Skromnost*	*50*
"tisto nekaj"	*50*	*Vodja*	*75*
		Hobiji	*25*
Pridobljeno	***Vpliv***	***Zunanji vpliv***	***Vpliv***
Izobrazba	25	*Standardi*	25
Strokovno znanje	25	Pogoji dela	50
Skupna stvarnost	50	*Število ljudi v skupini*	50
Skupen cilj	50	*Uporabniki*	100
		Obvestila	75

Ko prevedemo zahteve projektne skupine v poznane oblike razmerij, ugotovimo, da se pomen razmerij, ki najbolj vplivajo na izboljšanje kakovosti dela in s tem izdelka (uporabniške rešitve) giblje znotraj naslednjih razmerij:

1. Razmerja komunikacijske narave.
2. Razmerja koordinacijske narave.
3. Razmerja tehnične narave.
4. Razmerja tehnično motivacijske narave.
5. Razmerja kadrovske narave.

Ko razmerja in dejavnike, ki vplivajo nanje, vpnemo v grozd (Slika A.2), ugotovimo, da je v projektnih skupinah za razvoj uporabniških rešitev, poudarek

na ljudeh, na vsakem posamezniku in ustvarjanju njegovih povezav z ostalimi člani projektne skupine (Slika A.3). Povezave pa ustvarjamo tako s komuniciranjem kot tudi s standardi ter predpisi. Premik od organizacije nadzorovane samoiniciative k organizaciji vzpodbujene samoiniciative lahko po metodi, predstavljeni v [46, 43], ocenjujemo z izidom 60. Ta izid pomeni, da je za kakovostno ekipo za izdelavo uporabniške rešitve potrebna prijazna organizacija vzpodbujene samoiniciative [28].

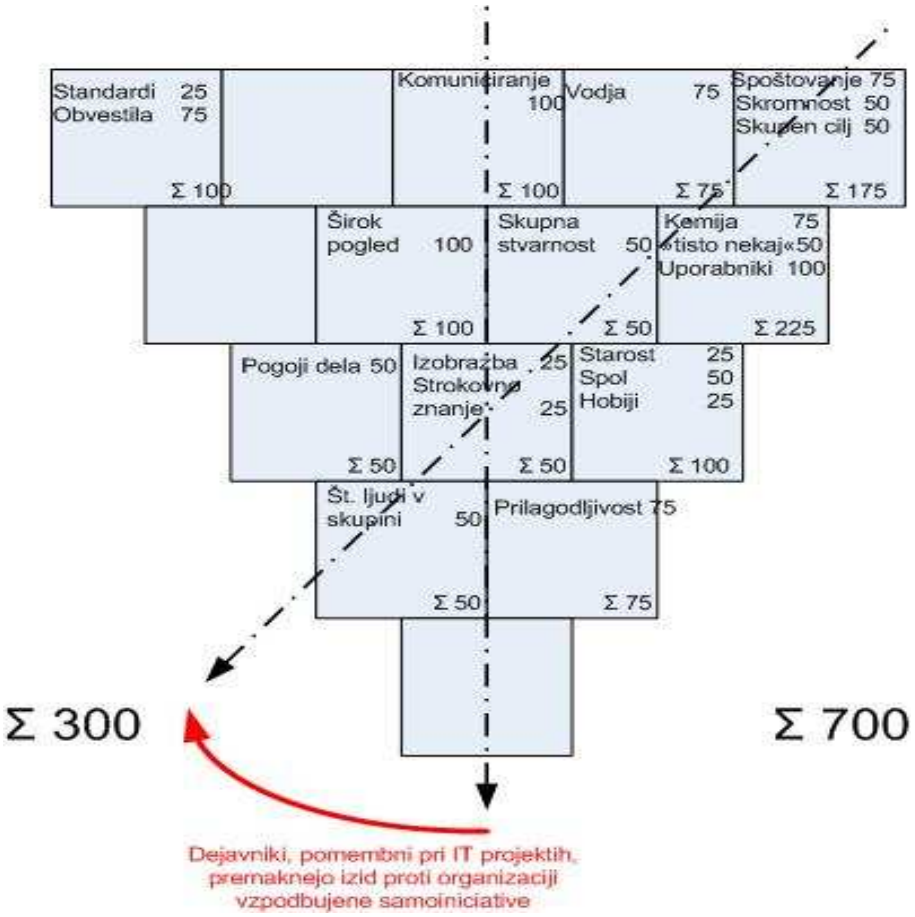

Figure A.3: Uteženost grozda razmerij z vplivnimi dejavniki in delni zasuk proti prijazni organizaciji

A.4.2 Predvideti uspeh

V podjetjih, ki se jih posredno dotikamo v tezi, programiranje uporabniških rešitev za podporo poslovanju ni osnovna dejavnost podjetja. Oddelki informatike predstavljajo manjši del zaposlencev v podjetju (povprečno okrog 5%).

Čeravno ljudje postajamo vse večji individualisti, pa se od nas zahteva skupinsko delo, delo na projektih in prenašanje znanja na sodelavce. Formalno zahteve pokrijemo z organizacijskimi predpisi, kultura sodelovanja, razmerja, ki jih ustvarjamo, pa so tisto, kar nas popelje h kakovostnim učinkom. Preglednost in znani cilji projekta so osnova na kateri člani projektne skupine gradijo sodelovanje. Če cilji niso poznani, pa je vsaka pot prava. V projektni skupini pričakujemo podrejenost zasebnih ciljev članov skupine skupnim ciljem in s tem privrženost združbi. Zahteva se poistovetenje s skupino, visoka raven skupnega napora, žrtvovanje in zaupanje. Skupina lahko postane ekipa in izoblikovanje ekipe je ob samem izdelku tudi naš projekt.

Nove uporabniške rešitve nastajajo v projektnih skupinah. Celotno poglavje 3 je posvečeno obvladovanju projektov (angl: *Project Management*). Razvoj vsake uporabniške rešitve ima pomembno lastnost projekta - je neponovljiv. Projekt omejujejo trije osnovni dejavniki: čas, viri (prvine) in stroški. Kot četrti dejavnik je v tezi vpeljana kakovost, tako projektnega dela, kot izdelka projektne skupine. Lastnost projekta je tudi, da imajo člani projektne skupine posebej dodeljene naloge, pristojnosti in odgovornosti.

Projektno skupino za razvoj nove uporabniške rešitve sestavljajo zaposlenci v informatiki, njihovi zunanji dobavitelji (največkrat programerji in ne tako pogosto analitiki procesov, saj je za uspeh projekta pomembno, da vsaj nekaj analitikov prihaja iz podjetja in ne od zunaj) ter uporabniki rešitve, ki izvajajo neko drugo funkcijo v podjetju oziroma so nosilci dela poslovnega procesa, ki ga bo rešitev podprla. Komuniciranje med njimi poteka tako formalno kot tudi neformalno. Pomembna so razmerja, od katerih so sodelujoči nekatera ustvarili že pred začetkom projekta, kar velja predvsem za zaposlence podjetja - naročnika.

Projektno skupino sestavi vodja projekta, tega pa določi lastnik projekta - notranji kupec. Pri dopolnjevanju skupine z zaposlenci iz podjetja naletimo na ovire, ki jih običajno postavljajo funkcijski vodje - nihče ne želi izgubiti najboljših ljudi. Zunanje sodelavce moramo največkrat pridobiti preko javnih razpisov. V splošno prakso pa žal ne sodi, da bi ene ali druge sodelavce izbirali na podlagi osebnosti, njihovega načina dela z drugimi sodelavci ali po kateremkoli v tabeli A.1 navedenih sodilih, razen morda po formalni izobrazbi in drugih strokovnih znanjih. Žal to ni dovolj za prehod skupine v ekipo, kar se zgodi, ko postane namen skupine razumljiv vsem članom in vsak odigra predpisano vlogo tako, da v največji meri uveljavi svojo nadarjenost in usposobljenost [46].

Projektni vodja je pred zahtevno nalogo, ko mora iz ljudi, danih mu na razpolago, potegniti najbolje in največ. Kateri so dejavniki, ki jih mora iskati in kje lahko vlaga manj svojih moči? Pri iskanju odgovora na vprašanje mu utegne biti v pomoč Tabela A.1. Projektna skupina, v kateri pomembni dejavniki niso navzoči v zadostni meri, je v veliki meri v naprej obsojena na neuspešno delo, združba pa na nekakovosten izdelek. Govorimo o mehkih

dejavnikih, kot jih pozna ravnanje z ljudmi (ang. *HRM - Human Resource Management*). Primerjaj z [38].

A.5 Zaključek

Metoda, ki jo uporabljamo v podjetjih, kot je podjetje, ki je bilo predmet raziskave za namen teze, je v resnici spremenjena in prilagojena "metoda Prilagodljivega razvoja programske opreme". To dejstvo spreminja formalnost našega dela. Negotovost pri razvoju ni več na sovražnik, postane nekaj, kar lahko uporabimo na pozitiven način. Ker so vhodni podatki za proces negotovi, je negotov tudi sam proces. Pravila, standardi in dokumentacija omejujejo in hkrati pomagajo projektnim skupinam pri razvoju uporabniških rešitev. Srečujemo se torej s precejšnjo negotovostjo glede trdnosti uporabniških zahtev na eni strani, na drugi strani pa z zelo formalnimi omejitvami pri naročilih in vsebini dokumentacije. To drugo posebej velja pri javnih razpisih v podjetjih v javni lasti. Naši zunanji dobavitelji pričakujejo natančno opredeljene zahteve, to pa pričakuje od nas tudi zakon.

Ker gre pri razvoju uporabniških rešitev za podporo poslovanja za delo multidisciplinarnih skupin iz različnih okolji, so značajske lastnosti posameznika v skupini do največ 10 članov v skupini izredno pomembne. Vendar ljudi z ustreznimi lastnostmi, ki sestavljajo projektno skupino, iz različnih vzrokov vedno ne moremo izbirati. Če pa jih poznamo, utegnemo iz ugodnih součinkovanj med lastnostmi sodelujočih potegniti najboljše učinke. Ljudje, ki delajo skupaj in uspešno komunicirajo z razvitim sodelovanjem dosegajo več kot, če uporabljajo samo individualne talente [14] . Zavedati pa se moramo, da ne moremo vedno delati z isto ekipo ljudi. Ljudje se eden drugega naveličajo, nove skupine pa so lahko tako nepredvidljive kot sam proces.

Sprašujemo se po razmerjih, ki nastajajo in se spreminjajo med ljudmi v projektni skupini za razvoj programske opreme. Upoštevali smo standarde, na področju kakovosti, projektnega vodenja, javnih naročil, zunanjih dobaviteljev, teorijo razvoja programske opreme in najboljše poznane prakse. Odgovor je ugodno medsebojno součinkovanje ljudi in zagotovitev predpostavk, ki do tega pripeljejo. Gre za sprožilce, ki v ljudeh povzročijo, da dajo še nekaj več kot običajno, ali ovire, zaradi katerih se "zaprejo" vase.

A.5.1 Potrditev hipotez

V nalogi sem empirično ugotovila, da so za uspeh dela projektnih skupin in njihovo preobrazbo v ekipe odločilnega pomena razmerja med člani skupine in s tem potrdila hipotezo št. 1. Razmerja so nekaj, kar ne moremo standardizirati in jih zato ne znamo v naprej določiti. Večinoma je formalno določena le zahtevana usposobljenost (npr. izobrazba) èlanov projektne skupine dejavnik osebnosti ljudi pa ni upo[1]tevan. Pri ugotavljanju pomena razmerij

smo uporabili metodo MUKOZ (glej poglavji 2.6 in A.4). Z njeno pomočjo smo določili razmerja, ki pomembno vplivajo na možnost sodelovanja v projektni skupini in izpostavili pomembne človeške (osebnostne in profesionalne) lastnosti ter lastnosti okolja, v katerem nastaja razvoj uporabniških rešitev. Te povratno vplivajo na razmerja v projektni skupini.

V gospodarskih javnih službah moramo žal izhajati iz predpostavke, da si ljudi za delo na projektih ne moremo izbirati niti znotraj podjetja niti pri pogodbeniku. Tako nam največkrat ostane možnost, da iz ljudi, določenih v skupino, napravimo ekipo le s pazljivim upoštevanjem različnosti vsakega posameznika in potrpežljivim ustvarjanjem sinergije med njimi.

Razmerja so bolj kot s človekovimi poklicnimi, pogojena z osebnimi značilnostmi sodelujočih. Idealna kombinacija osebnih in poklicnih značilnosti omogoča idealna razmerja v projektni ekipi, katere organizacija je po teoriji [43] že blizu značilnostim prijazne organizacije. To je bila vsebina hipoteze št. 2.

S tretjo hipotezo smo preverjali primernost metode prilagodljivega razvoja programske opreme razmeram v podjetjih z značilnostmi ELESa. Metoda izhaja iz tega, da so spremembe edina stalnica v današnjem poslovanju, zato predvidi tudi razdelitev vlog, določenost cilja in omejitve v času. V ospredje postavi človeka, kot tistega, ki se prilagaja. Prilagajajo (preurejajo) pa se tudi razmerja med ljudmi v projektni skupini. Verjetno obstajajo tudi drugi načini ali metode za doseganje kakovostnih rezultatov pri razvoju uporabniških rešitev, a v poznanem okolju ELESa, kot v primerljivh poslovnih okoljih, se je metoda prilagodljivega razvoja programske opreme pokazala kot ustrezna.

Bibliography

[1] Abrahmson Pekka, Salo Outi, Ronkainen Jussi & Warsta Juhani. *Agile software development methods*, VTT Publications 478, 2002

[2] Adams Douglas. *The Hitchhiker´s Gide to the Galaxy*, Serious Productions Ltd., 1995

[3] Artto Karlos A. Management Across the Organisation, *Project Management*, Volume 5 (1), 1999

[4] Bajec Marko, Krisper Marjan. Agilne metodologije razvoja informacijskih sistemov, *Uporabna informatika*, letnik XI (2), 68-76, 2003

[5] Bajec Marko, Krisper Marjan. A methodology and tool support for managing business rules in organisations, *Information Systems,* 30, 423-443, 2005

[6] Bandelj Franjo. *Magistrska naloga*, Univerza v Ljubljani, Ekonomska fakulteta, 2004

[7] Ben-Menachem Mordechai and Garry S. Marliss. *Software Quality - Producing Practical, Consistent software,* International Thomson Computer Press, Boston, 1997

[8] Bogue Robert. Undestanding the Different Goals of Software Development, /www.developer.com/, 2004

[9] Bowen David E. Interdisciplinary Study of Services: Some Progress, Some Prospects, *Journal of Business Research*, Volume 20 (1), 1990

[10] Bowen John, Ford Robert C. Managing Service Organizations: Does Having a "Thing" Make a Difference?, *Journal of Management*, 28 (3), 447-469, 2002

[11] Brezavšček Alenka, Zupan Lucija. Standardi in priporočila na področju informacijske varnosti, Dnevi slovenske informatike, 2006

[12] Brooks Frederic P. *The Mytical Man-Month,* Adisson-Wesley, 2002

[13] Clegg Stewart R., Pitsis Tyrone S., Rura-Polley Thekla, Marosszeky Marton. Governmentality Matters: Designing an Alliance Culture of Inter-organizational Collaboration for Managing Projects, *Organization Studies*, Volume 23 (3), 317-337, 2000

[14] Cockburn Alistair, Highsmith Jim. Agile Software Development: The People Factor, *Computer*, November, 2001

[15] Cockburn Alistair. *Agile Software Development*, Addison-Wesley, Boston, 2002

[16] Dennis Alan, Wixom Haley Barbara, Tagareden David. *System Analysis and design - An Obejct -Oriented Approach with UML*, John Wiley & Sons, 2002

[17] Dworatschek Sebastian, Meyer Helga. Competence and qualifcation requirements of project personnel, http://www.ipmi.uni-bre.de, 2001

[18] Fowler Martin with Scott Kendall. *UML Destilled*, Addison-Wesley, Boston, 1997

[19] Fowler Martin. The New Methodology, /www.martinfowler.com/, 2003

[20] Gamma Erich, Helm Richard, Johnson Ralph, Vlissides John. *Design Patterns*, Addison-Wesley, Reading, 1994

[21] Goolsby Kathleen. The Snowball Effect: Characteristics of Outstanding Outsourcing Relationships, White Paper by Outsourcing Center, 2002

[22] Greif N., Richter P. Software engineering related standards and guidelines for metrology, submitted to *World Scientific*, 04.06.1999

[23] Highsmith James A. III. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, New York, 2000

[24] Kerzner Harold. *Advanced Project Management - Best practices of implementation*, John Wiley & Son, Ohio, 2004

[25] Kidder Tracy. *The soul of the new machin*e, Modern Libary Edition, 1997

[26] Kislik Felix A. *Building Long Term Relationship*, Trestle Group, 2004

[27] Kolar Alenka. Tehnični in organizacijski izzivi pri integraciji informacijskih sistemov - primer ELES, SLOKO CIGRE - CIRED, 2005

[28] Kolar Alenka. Vpliv razmerij v projektni skupini na kakovost uporabniške rešitve, Dnevi slovenske informatike, 2006

[29] Kolar Alenka, Čater Jure. Can managing the outsourcing on different projects be creative?, NORDNET Project Management Creativity, Stockholm, 2001

[30] Kolar Alenka, Čater Jure. Informacijski projekti - organizacijski problemi, Združenje za projektni management - Projektni forum, Maribor, 2003

[31] Kolar Alenka, Lagler Boris. e-projects: A kingdom for a horse, IPMA World Congress, Budapest, 2004

[32] Kolar Alenka, Lagler Boris. Projects in Elektro Slovenija (ELES): we did it our way, NORDNET, Helsinki, 2004

[33] Kolar Alenka, Lagler Boris. Enterprise Application Integration - Why business needs process?, CIGRE Colloquium SC D2, Cuernavaca, Mexico, 2005

[34] Kolar Alenka, Nemec-Pečjak Marko. Requirement to standardised project work in the field of information technology in Elektro-Slovenija d.o.o., SENET, 2000

[35] Krisper Marjan, Rupnik Rok, Bajec Marko, Rožanec Alenka, Zrnec Aljaž, Vavpotič Damjan, Osojnik Rok, Tomažič Roman. Enotna metodologija razvoja informacijskih sistemov (EMRIS), Vlada Republike Slovenije, Center vlade RS za informatiko, 2003

[36] Lewis, E.E. *Introduction to relaiability engineering*, John Wiley & Sons, New York, 1987

[37] Linstead Stephen, Fulop Liz, Lilley Simon. *Managemnt and organization - A critical tekst*, Palgrave Macmillan, New York, 2004

[38] Lipičnik Bogdan. *Ravnanje z ljudmi pri delu (Human Resource Management)*, Gospodarski Vestnik, Ljubljana, 1998

[39] Lipovec Filip. *Teorija organizacije*, Univerza v Ljubljani, Ekonomska fakulteta, 1974

[40] Marshall Chris. *Enterprise Modelling with UML - Designing Successful Software through Business analysis*, Addison Wesley Longman, Inc., chapter 3, 63-73, 2000

[41] McCall, J.A. Quality factors in J.J. Marciniak (ed.) *Encyclopedia of Software Engineering*, John Wiley and Sons, New York, 1944

[42] McConnell Steve. *Rapid development*, Microsoft Press, Washington, 1996

[43] Mihelčič Miran, Bračko Cita, Gabrijelčič Janez, Kline Miro, Šček Janez, Štucin Ivan. Metodologija ugotavljanja kakovosti ali popolnosti organizacije (gospodarskih) združb (Methodology of assuming the quality of firms´ organization), research paper, Chamber of Economy of Slovenia, 101 - 129, 1988

[44] Mihelčič Miran. *Organizacija in ravnateljevanje*, Založba FE in FRI, Ljubljana, 2003

[45] Mihelčič Miran. *Poslovne funkcije (Business Functions)*, Založba FE in FRI, Ljubljana, 2004

[46] Mihelčič Miran. Principles of Organization Analysis; Sugestion of Method and Application in Practice, EGOS Library, 2004

[47] Moss Kanter Rosabeth. Collaborative Advantage: The Art of Alliances, *Harvard Business Review*, July - August, 1994

[48] Norvig Peter and Cohn David. *Adaptive Software*, Harlequin Incorporated, 2004

[49] Nemec-Pečjak Marko. *Hitri vodnik skozi Microsoft Project 98*, Založba Atlantis, Ljubljana, 1998

[50] Nemec-Pečjak Marko, Kolar Alenka, Kelšin Drago. Projektni informacijski sistemi in Oracle 8i, SIOUG- 5. strokovno srečanje uporabnikov programske opreme, Portorož, 2000

[51] Noteboom Bart, Berger Hans and Noorderdhaven Niels G. Sources, Measurement and Effect of Trust in the Governance of Buyer-Supplier Relations, Research project, Bruge, 2004

[52] Outsourcing center. Governing Attitudes: 12 Best Practices in Managing Outsourcing Relationships, May 2002

[53] Reich Horner Blaize. Using knowledge management to improve IT project success, IPMA World Congress, Budapest, 2004

[54] Reich Horner Blaize. Managing Knowledge within IT Projects: A Review of Current Literature and Directions for Future Research, working paper, 2004

[55] Ropponen Janne and Lyytinen Kalle. Components of software development risk: How to address them? A project management Survey, *IEEE Trans. on Software Engineering* 26 (2), 98-112, 2000

[56] Rosqvist Tony, Koskela Mika and Harju Hannu. Software Quality Evaluation Based od Expert Judgement, *Software Quality Journal*, 11, 39-55, 2003

[57] Sandström Erik. EAI Development at Vatenfall, CIGRE Colloquium SC D2, Cuernavaca, Mexico, 2005

[58] Sindermann, Carl J. *Winning the Games Scientist Play*, Plenum, New York, 1982

[59] Solina Franc. *Projektno vodenje razvoja programske opreme*, Založba FE in FRI, Ljubljana, 1997.

[60] Starc Alenka. Cenitev projektov razvoja programske opreme, *Magistrsko delo*, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 2004

[61] Škufca Matej. Postopek oblikovanja hiše kakovosti, *Diplomska naloga*, Univerza v Ljubljani, Fakulteta za strojništvo, 1995

[62] Štubljar Milan. Vodenje majhnih skupin in majhnih projektov, Univerzitetno diplomsko delo, Univerza v Ljubljani, Fakulteta za računalnitvo in informatiko, 2006

[63] Takeishi Akiro. Bridging inter- and intra-firm boundaries: Management of supplier involvement in automobile product development, *Strategic Management Journal*, 22, 403-433, 2001

[64] Trček Bogdan. Projektna pisarna 2.3., Dnevi slovenske informatike, 273-276, 2002

[65] Tuohey William G. Benefits and Effective Application of Software Engineering Standards, *Software Quality Journal*, 10, 47-68, 2002

[66] Vavpotič Damjan, Bajec Marko, Krisper Marjan. Measuring and improving software development methodology value by considering technical and social suitability of its constituent elements, working paper, 2005

[67] Vavpotič Damjan, Bajec Marko, Krisper Marjan. Conceptual Model for Socio-technical evaluation of SW development methodologies, Dnevi slovenske informatike, Portorož, 2005

[68] Walgenbach Peter. The Production of Distrust by Means of Producing Trust, *Organisation Studies,* 22 (4), 693-714, 2001

[69] Walz D. B., Elam J. J. and Curtis B. "Inside a Software Design Team: Knowledge Acquisition, Sharing and Integration." *CACM* 36 (10), 63-77, 1993

[70] Yeo K.T. Critical failure factors in information system projects, *International Journal in Project Management,* 20, 241 - 246 , 2002

[71] Zrnec Aljaž, Bajec Marko, Krisper Marjan. Business modelling with UML, Zajc Baldommir (Ed.), Zbornik desete Elektrotehniške in računalniške konference ERK 2001, 24-26, IEEE Region 8, Slovenska sekcija IEEE, 31-34 , 2001

[72] Živkovič Aleš. Metoda za določanje obsega objektno-orientiranih programskih projektov, *Doktorska disertacija*, Univerza v Mariboru, Fakulteta za elektrotehniko in računalinistvo, 2004

[73] A Guide to the Project Management Body of Knowledge, Project Management Institute, 1996

[74] ANSI/IEEE standard 730.1, Standards for Software Quality Assurance Plans, 1989

[75] Applied Marketing Science Inc., http://www.ams-inc.com/whatwedo

[76] Client Advisor, Summary of META Group's Weekly Research Meeting, 28 January 2003

[77] Energetski zakon, Uradni list Republike slovenije, št. 79/1999

[78] *Galup Managemet Journal* - WEB page, www.gmj.gallup.com, May 2006

[79] ISO/IEC 6592 Information technology - Guidelines for documentation of computer based application systems, 2000

[80] ISO/IEC 9294 Information technology - Guidelines for the management of software documentation, Geneve, 1990.

[81] ISO 10006 Quality management systems - Guidelines to quality management in projects, 2003

[82] ISO/IEC 12207 Information technology - Software life cycle processes, 1995

[83] ISO/IEC TR 15846 Information technology - Software life cycle processes - Configuration Management, 1998

[84] ISO/IEC 15910 Information technology - Software user documentation process , 1999

[85] ISO/IEC 20000 Information technology - Service Management, 2005

[86] Internal committee draft of international standard, software user documentation process ISO/IEC JTC1/SC7 WG2, draft at 2000-01-04

[87] ITIL Information Technology Infrastructure Library, The key to Managing IT Service, Best Practice for Service Delivery, Office of Goverment Commerce, London, 2001

[88] Operation Excellence Newsletter from METAGroup, 17.jan.2003

[89] PSIST ISO/DIS 9000, Sistem vodenja kakovosti - Zahteve, 3rd Eddition, May 2000

[90] Rational Unified Proces, Best Practices for Software Development Teams, Rational Software White Paper, TP026B, Rev 11/01

[91] Report from the Gartner Group Lecture "Poslovna vrednost informacijskega sistema - V iskanju svetega grala", Ljubljana, 4.7.2002

[92] RTCA/DO-178B. *Software Consideration in AirbornSystems and Equipment Certification*, Washington, RTCA Inc., 1992

[93] TickIT 5.0 Executive Overview, http://www.tickit.org/overview.pdf., 2000

[94] Zakon o javnih naročilih (Public Procurement Law), Uradni list Republike Slovenije št. 39/00, 2000

Izjava

Spodaj podpisana Alenka Kolar izjavljam, da sem doktorsko nalogo izdelala samostojno, pod vodstvom mentorja prof. dr. Franca Soline.

Ljubljana, 01.06.2006

Alenka Kolar

Zahvala

Zahvaljujem se prof. dr. Francu Solini za potrpljenje, nasvete in neposredno pomoč.

Moji mami in očetu, ker sta ravno prav sitna in me imata tako rada, da človek vztraja in se želi dokazati. Ker mi skrbno pregledujeta moje tekste in ne obupata nad menoj. Ker vedno poskrbita, da se da in da lahko.

Borisu, ki mi ni pustil obupati, Roku, ki je hodil sam spat, da je mama lahko delala, Marjanu, ker je pomagal z nameščanjem programske opreme, Juretu in Boštjanu za pomoč pri "energetskem delu" teze, Timu in Aleksu za pomoč z LaTeX-om, Jovanu in Igorju za vzpodbudo in nasvete, Marku, ker mi je pokazal svet projektnega vodenja.

Sodelavcem Službe za obratovanje in razvoj uporabniških rešitev (Aleks, Ana, Jernej, Luka, Maja, Marko, Matjaž, Mojca, Monika, Vladimir), ker so pokrivali mojo odsotnost z dela in mi priskrbeli temo za doktorsko disertacijo. Vsem mojim direktorjem, ker so mi študij omogočili.

Zahvaljujem se tudi vsem, ki jih nisem posebej imenovala in ki so na mnoge načine prispevali k nastanku tega dela.