Master's Thesis

Developing a document management system for Infineon using i* for requirements engineering

Marco Lautischer, BSc
m.lautischer@student.tugraz.at
Matr. No. 0631079

Institute for Knowledge Management
Graz University of Technology

Graz University of Technology

Supervisor: Dipl.-Ing. Dr. techn. Markus Strohmaier

Klagenfurt, March 2012

Abstract

Creating and introducing new software is a tedious process where plenty of things can go wrong. Therefore every step has to be well-considered and designed in detail. Particularly, if several stakeholders with different requirements are involved in the creation process, initial efforts in requirements engineering are absolutely essential to come to a successful project outcome.

The topic of this master's thesis was advertised by the Infineon Technologies Austria AG, to improve the management of their "Test Engineering Processes". The following chapters demonstrate the requirements engineering process using the i* Framework - an agent-oriented approach - to specify, design and create a document management system for Infineon. The main challenges and appropriate solutions appearing from project start until the software is used inside the company are addressed. The realization of the system is also presented, where the requirements of different departments, as well as of various sites are considered. The practical usage of i* to come to terms with this problem is based on the theoretical input of previous work. This framework is initially used to determine the stakeholders and their requirements on the software with the assistance of graphical models. Afterwards different implementation possibilities are analyzed and evaluated to find the best-fitting technology. This thesis also provides some background knowledge on requirements engineering and the i* Framework with its notation.

Keywords: Requirements Engineering, i*, i* Framework, Document Management System, Strategic Dependency Model, Strategic Rationale Model

Kurzfassung

Die Erstellung und Einführung einer neuen Software ist ein langwieriger Prozess, bei dem Vieles schief gehen kann. Deshalb muss jeder Schritt ordentlich durchdacht und ausführlich geplant werden. Sind mehrere Interessensvertreter mit unterschiedlichen Wünschen in den Erstellungsprozess involviert, so ist die Durchführung einer Anforderungsanalyse absolut notwendig, um das Projekt erfolgreich abschließen zu können.

Die Aufgabenstellung dieser Masterarbeit wurde von der Infineon Technologies Austria AG ausgeschrieben, um die Verwaltung interner Testprozesse zu verbessern. Die folgenden Kapitel demonstrieren den gesamten Prozess der Anforderungsanalyse unter Verwendung des i* Frameworks - einer agenten-orientierten Herangehensweise - um ein Dokumentenmanagementsystem für die Firma Infineon zu spezifizieren, planen und umzusetzen. Die größten Herausforderungen vom Projektstart bis zur Verwendung der Software innerhalb der Firma, werden angeführt und entsprechende Lösungen präsentiert. Weiters wird die Realisierung des Systems, die sämtliche Vorstellungen der unterschiedlichen Abteilungen und sogar anderer Firmenstandorte berücksichtigt, dargestellt. Die praktische Verwendung von i* für Lösungen dieser Problemstellung basiert auf dem theoretischen Input bisheriger Arbeiten. Dieser Framework wird zunächst verwendet, um mit Hilfe von grafischen Modellen sämtliche Interessensvertreter und deren Anforderungen an die Software zu identifizieren. Anschließend werden damit unterschiedliche Implementierungsmöglichkeiten analysiert und evaluiert, um dadurch die geeignetste Technologie zu finden. Diese Arbeit liefert außerdem einige Hintergrundinformationen über die Thematik der Anforderungsanalyse und das i* Framework bzw. deren Notation und Verwendung.

Schlagworte: Anforderungsanalyse, Requirements Engineering, i*, i* Framework, Dokumentenmanagementsystem, Strategic Dependency Model, Strategic Rationale Model

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted, either literally or by content from the used sources.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Klagenfurt	13.03.2012	[illegible]
Ort	Datum	Unterschrift

Acknowledgments

At this point I want to thank to my advisor. Dipl.-Ing. Dr. techn. Markus Strohmaier supported my idea of an industrial master thesis at Infineon Technologies Austria AG as an excellent supervisor and for suggesting me such a perfect fitting theoretical background. During the entire process he gave me answers to my questions and offered assistance and useful ideas and proposals.

I also want to thank my supervisor at the Infineon Villach, Johann Brandner, and the internal leader of this project, Mario Nußbaumer for giving me support and providing the needed information to plan and realize this project. Additionally, my gratitude goes to all colleagues, that amused my working days and gave me a good time at the company.

Last but not least I would like to express my gratitude to my family for giving me the possibility and financial support to study at the University of Technology in Graz and of course my girlfriend Claudia for motivating me and being there for me at any time.

Klagenfurt, March 2012 Marco Lautischer

Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
CSS	Cascading Style Sheets
DMS	Document Management System
FR	Functional Requirements
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
JSP	Java Server Pages
KM	Knowledge Management
NFR	Non-Functional Requirements
RE	Requirements Engineering
SDM	Strategic Dependency Model
SRM	Strategic Rationale Model
SRS	Software Requirements Specification

1. Introduction

1.1. Motivation

Infineon Technologies is one of the largest producers of semi-conductors all over the world. The location in Villach is in charge of production and development. Here, among other things, the test development is situated, to which a lot of attention is given by the company. This division accomplishes predefined test processes for a huge amount of projects. Each test process consists of different milestones that are processed sequentially. To finish one milestone, several important files need to be created, finished and reviewed first. For project leaders, it is essential to know the actual status of each project at any time.

For many years, the Infineon Villach has stored all information about its "Test Engineering Processes" in different folders on network hard discs. Changes to these files get distributed among the integrated and entitled persons per email or are stored in shared directories. Because of the huge amount of data and differences in folder structure, folder names, file names etc. it is hard to retrieve needed information. The different file-versions appearing during one test process complicate the clarity of the project information and the assignability of files to milestones.

Because of these facts, a new file management of such test-engineering-specific information is needed that includes functionality for information search and retrieval by offering a simple user interface, that can easily be used by all employees without a big effort.

Such central file management brings benefits for everyday work by reducing lost time and frustration caused by not finding the right information or data loss. Furthermore, it reduces double work and improves the cooperation within one location and between the different facilities of the company.

The development of such huge software must be handled with care and initial efforts in planning and design are needed to make it usable and reliable. Requirements engineering - or short RE - is part of the analysis- and design phase and helps to identify the scope and framework conditions of the project at initial state and before the implementation starts. This affects the project-success in a positive way and avoids expensive and time-consuming changes in late development phases. Another reason for the increasing importance of requirements engineering is the growing relevance of software in almost every branch of industry and the resulting competition to high quality [Partsch, 2010, p. 6ff].

1.2. Objective

The initial steps of the project are lead by the project vision, that is defined by [Brandner, 2011] as follows:

> *"Every time a user needs information or documents about one specific project, he opens an intranet site or a simple tool, where he can easily find it. Changes do not get distributed per mail or in a shared directory, as it is done now, but only with the new software. The acceptance of this system is one of the main goals and there must not exist a comparable tool."*

The contribution of this master thesis is the whole process of realizing this loose and undefined vision. Following steps are included:

- Introduction to knowledge management (KM): a short theoretical introduction to the term *knowledge management* is given to the employees of the test development division, that are called "test engineers". This provides an overview of the different architectures and techniques in KM and shows different possibilities on how such visions can be realized. This input also helps to recognize the current situation and to improve communication and understanding throughout the whole development process.
- Requirements engineering (RE): to specify how to realize this vision, the requirements are acquired first. Therefore the requirements engineering process needs to be walked through, where stakeholders' wishes and demands are figured out. The gathered information is visualized with drawable elements defined in the i* Framework that offers an agent-oriented approach to the requirements engineering process. Two different i* models, created with the open source software "Open OME" can be seen as a visualization of the stakeholders' requirements on the software and the different dependencies among themselves [Yu and Liu, 2000].
- Technology choice: the choice of the best-fitting technology is one important step in this project. This can be found out with the help of the

created i* models: the strategic dependency model that defines the requirements, and the strategic rationale model that expands the first model by internal rationales to obtain a high detail level and better analysis possibilities. After goal evaluation, different labels are assigned to the i* elements [Yu and Liu, 2000]. They ease the technology choice by adding a status to each node [Yu and Liu, 2000, p.7]:

  - satisficed
  - partially satisficed
  - conflicted
  - unknown
  - partially denied
  - denied

- Specification of the software: before the software is implemented, a software specification is created, that describes useful libraries and the different components of the system with its structures. It gives an overview of the functionality of the tool and acts like a guideline during the whole fulfillment process. It also defines the underlying conditions and acts like a book of reference if questions arise.
- Implementation: the software is implemented according to the specification taking into account internal restrictions.
- Testing: before the system is introduced, it is tested for a certain amount of time. This avoids bugs and other problems appearing during usage. It is very important that the software is able to run for a long period of time without any additional effort and with little maintenance.
- Introduction of the system: the last part of this work is the introduction of the system, where new possibilities are presented to the users and how they can use the functionality of the software.

1.3. Thesis Outline

This work is structured mainly into six chapters.

- Chapter 1 discusses the motivation for this master thesis. This includes introducing the company that advertised for this project and stating the actual situation and problems that need to be solved. An overview of all the steps that are performed to plan and design a new software-based system - in this case a document management system - within a company, is given.
- Chapter 2 gives a theoretical introduction into the main topics dealt with in this work and presents examples of the most relevant literature in the context of this thesis. First the term *requirements engineering* is described, the advantages are shown and the approach is listed. Other important terms are defined here. Then the i* Framework, its notation in form of drawable elements with certain rules and its usage are presented. A small practical example helps to understand how the different models can be created and how they improve the identification of stakeholders and requirements. The last topic gives general information about document management, the different possibilities to create such a system and its benefits.
- Chapter 3 presents how the process from gathering the requirements from stakeholders up to the creation of the different models with i* is performed. The reader is informed how the implementation-technology is chosen by showing and describing the created models and the performed goal analysis. The selection is also justified.
- Chapter 4 details the implementation of the software by describing how the chosen technology is used and how the tool is structured. Some figures help to understand the interaction of different components and the communication between them.
- Chapter 5 shows the output of the project. This is done by analyzing and describing the finished software and comparing it with the distinguished requirements. The reader gets an impression of how far the requirements of the different stakeholders are fulfilled and what requirements are not satisfied at all.
- Chapter 6 sums up the performed work and describes statements and satisfaction of the involved parties in form of a conclusion. Problems during the development can also be seen here. Ideas for potential extensions and improvements are given.

2. Related Work

This chapter provides a theoretical introduction into this practical project by giving background information on the following topics:

- Requirements Engineering
- The i* Framework
- Document Management Systems

For each topic, the relevance for this project is highlighted and important literature is presented.

2.1. Requirements Engineering

2.1.1. Introduction

Constructing software-based systems desires a lot of planning and design to restrict and avoid mistakes or bugs that produce problems and harm the software. Because of that fact, software-engineering became an own discipline during the 60s. This discipline focused on the development of reliable software with high quality, low costs and a predictable delivery date. At that time different process models were explored and introduced [Partsch, 2010, p.1ff]. The first model is shown in figure 2.1.

The waterfall model is categorized into different phases; the first one - "Analysis and Definition" - serves the discovery and the clarification of requirements and furthermore the goal setting. This model is used in the defined order, in which all phases are performed after another with only few or no iteration. Boundaries are not hard and sometimes phases overlap, but the approach remains the same [IEEE, 1990, p.81]. The simple model does not completely correlate with reality and in practice its properties often cannot be kept [Partsch, 2010, p.3ff]. That is a reason why today many different approaches like incremental development,

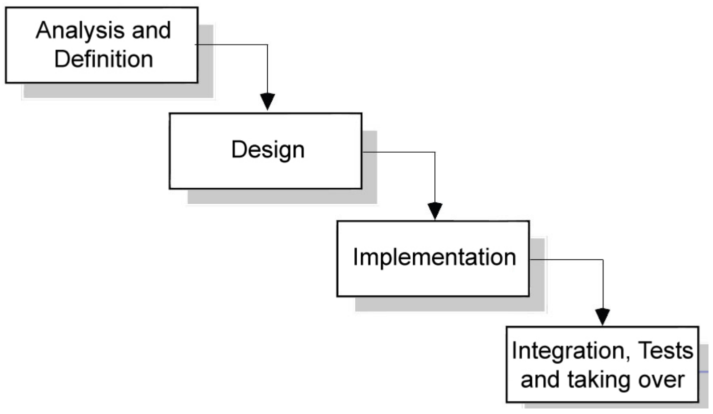


Figure 2.1.: Structure of the Waterfall Model (adapted from [Partsch, 2010, p.3]).

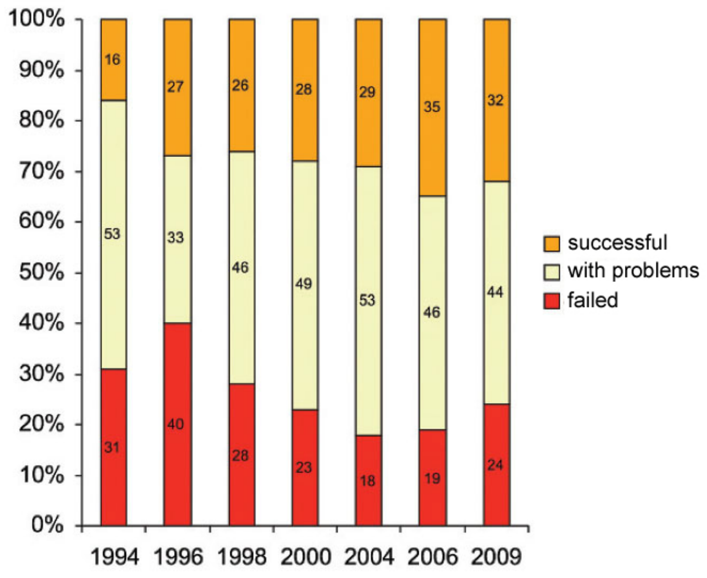


Figure 2.2.: Success in software projects (adapted from [Partsch, 2010, p.5]).

rapid prototyping or the spiral model, defined by the IEEE (The Institute of Electrical and Electronics Engineers) in [IEEE, 1990], exist.

All the effort spent in the software engineering discipline, the analysis and discovering of engineering processes and the introduction of different models over the last four or five decades could not eliminate the technical and economic risks of planning and realizing extensive software-based systems. Figure 2.2 shows that the amount of successful IT projects is smaller than one third, what is a considerable small amount [Partsch, 2010, p.5f].

[Becker and Huber, 2008] analyze the low success rate of about one third in IT projects. The authors see the reason why projects fail rather in the so-called "soft facts" than in the development technology. Their study focuses on the elicitation of different problems appearing in projects. The participants come from different working environments and play different roles in projects. The percentage distribution of quoted roles is shown in figure 2.3.

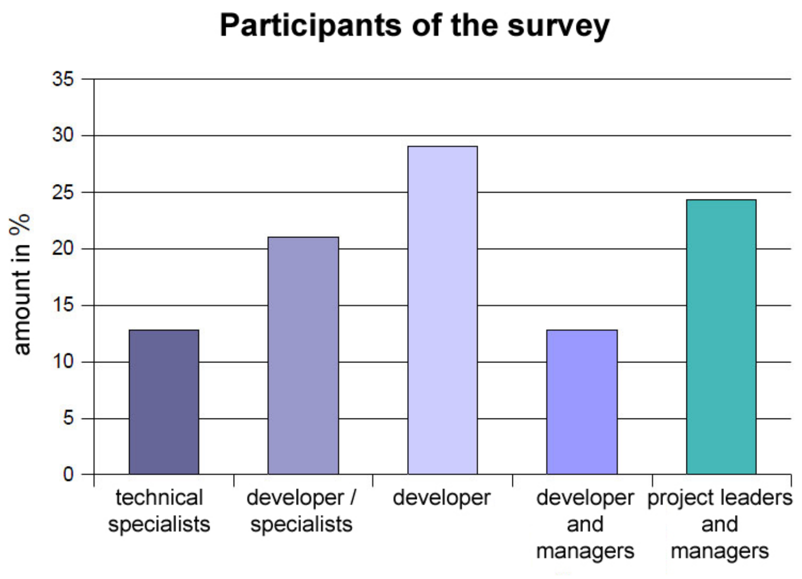


Figure 2.3.: Role distribution in projects (adapted from [Becker and Huber, 2008, p.4]).

The poll rating confirms the assumption of the authors and states that the biggest influence on project failure is given by partially unknown targets or intentions and a lack of information as well as communication. Other risks appear within the size of project teams and the missing trust between the team members.

Chosen technologies or procedure models are not the trigger for the bad outcome [Becker and Huber, 2008, p. 8ff].

[Partsch, 2010] states that about half of the problems appearing in software projects are caused by aspects that can be assigned to requirements engineering. Examples of such aspects are listed below [Partsch, 2010, p.6ff]:

- Insufficient end-user integration: users should be involved at the beginning of the development process, which leads to a user-centered design and implementation and avoids complex and expensive changes later on.
- Incomplete requirements: during the realization process, requirements often change or new requirements appear. This is a well-known problem but frequently too little attention is paid to it.
- Unclear objectives: it is problematic if the target is not clear and it is not exactly known what actually should be reached by a certain project. Unclear targets lead to incomplete requirements and this has a negative effect on efficient system design. Subsequent changes in requirements and a booming effort are the results.
- Inherent complexity of the task in hand: high complexity is caused by a big range of available information, the unrealistic expectations of the contracting authority and the diverse relationships between the different components. Each restriction, added by appropriate stakeholders, increases the complexity level.
- Communication: communication is - in the face of the huge amount of available communication technologies - still problematic, difficult and initiator of many problems. That is because of the different types of people with dissimilar background knowledge, fields of interest and roles, working together at the start of a project.

An additional problem, according to [Partsch, 2010], is the mutual dependency and reliability between those aspects. The existence of one such negative fact often results in the support of another one. As an example, the interdependency between complexity and clear targets is used: the bigger the complexity, the harder it is to make a complete and precise description which then can lead to unclear objectives [Partsch, 2010, p.6].

Finding and constructing a satisfying solution for the above mentioned problems is the main goal of all efforts in requirements engineering, that include, amongst others, the following actions [Partsch, 2010, p.7]:

- Stronger involvement of all stakeholders.

- Clear separation of concerns and responsibility.
- Usage of an adequate notation to visualize precise, consistent and complete requirements.
- Identification of key requirements that are written down. Some space for later changes should be planned here.
- Taking the whole development process into account within the requirements engineering scope.

Requirements engineering has a big influence on the development process as for example mentioned by [Brooks, 1987, p.8] in an article that says [Partsch, 2010, p.7]:

> *"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."*

So, it can be said that requirements engineering is a kind of key phase in the development process of software-based systems. If it is performed efficiently, it brings benefits by trying to reduced the nearly 4/5 rejection and reworking costs from projects that are caused by mistakes in requirements engineering [Partsch, 2010, p.7f].

2.1.2. Definitions

Some of the new words and technical terms that were used in the introduction have not been given a description yet. In this subsection those expressions are precisely defined and characterized for a better understanding.

Requirements

Requirements are defined in a variety of ways. The IEEE lists the following three meanings of the word *requirement* in [IEEE, 1990, p.62]:

> *"(1) A condition or capability needed by a user to solve a problem or achieve an objective.*
>
> *(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.*
>
> *(3) A documented representation of a condition or capability as in (1) or (2)."*

In [Partsch, 2010] requirements are described as statements about tasks and performances to be fulfilled by a certain system or product, a process or a person, involved in this process. These statements contain answers to some questions [Partsch, 2010, p.25]:

- *Why is a certain system needed?*
- *What is the scope of services, the system should include?*
- *Are there restrictions that need to be observed?*

Several aspects are used to classify requirements. The distinction between functional- and non-functional requirements is the most important one and mandatory for this project.

Functional requirements describe *what* a system or a software should be able to perform, according to the conceptual formulation. In other words, the behavior of that system is described by defining how to act on certain inputs [Chung et al., 2000, p.6].

Non-functional requirements (NFR) describe *how* a system performs its tasks. This is in any case harder to check (than the *what*) because it cannot be tested automatically. Each non-functional requirement needs to be evaluated separately. Amongst others, requirements on the software performance and the external interface, restrictions in design and quality criteria are included here [Chung et al., 2000, p.13f].

A listing of classification criteria for requirements and a description can be found in [IEEE, 1998a, p.6].

Requirements Engineering

According to [Partsch, 2010] the term *requirements engineering* can be described as the sum of activities that need to be performed at the beginning of a project, usually in the "Analysis- and Design phase". The target of these initial steps is the elicitation, definition, description and analysis of all requirements on the system that needs to be designed and implemented. The three most important subtasks in requirements engineering are listed as follows [Partsch, 2010, p.20f]:

- Determination of requirements
- Documentation of requirements
- Analysis of the requirements description

The process of analyzing the requirements is also defined in [IEEE, 1990, p.62f] as:

> *"The process of studying user needs to arrive at a definition of system, hardware, or software requirements"* and *"The process of studying and refining system, hardware, or software requirements."*

[Garlan, 1994] gives another perspective on the term *requirements engineering.* The author describes the role of software architecture in requirements engineering as a correlation of problem space and solution space. While requirements engineering - which aims at the determination of the problem-dimensions - can be assigned to the first term, software architecture - which tries to design the structure of a solution out of a set of requirements - applies to the second one. So the dynamic relation between the problem-identification and the specification of its solution needs to be understood.

Software Requirements Specification

The software requirements specification (SRS), or often described as requirements document is created at the end of the requirements engineering process and contains the results of this process. Here, all the acquired requirements to a system are written down with a detailed and precise description. The content of such document is listed in [Partsch, 2010, p.32ff], where also quality criteria are quoted. A further definition to the SRS, including a schedule of properties for all the requirements in this document, is given by [IEEE, 1998a, p.4].

A recent approach in performing requirements documentation is described in [Brill et al., 2010]. The authors propose the use of ad-hoc videos as an additional representation, particularly for early requirements. Their study deals with the comparison of videos and textual representation under time pressure. The advantage of this new way of documentation is a clearer impression on the functionality of a system, where the usage is easier to understand. This helps the stakeholders to give better feedback and avoids misunderstanding in the initial steps of the project.

[de Bruijn and Dekkers, 2010] show the problems of the ambiguity in software requirements that are documented in natural language. The case study - in which a huge project that takes about 21 man years is analyzed - detects, that less than 10% of the requirements are totally clear. The resulted ambiguity of the evaluated requirement categoreis is visualized in figure 2.4. Table 2.1 lists the 15 categories, in which can be seen, that too less effort is given to avoid unclear requirements.

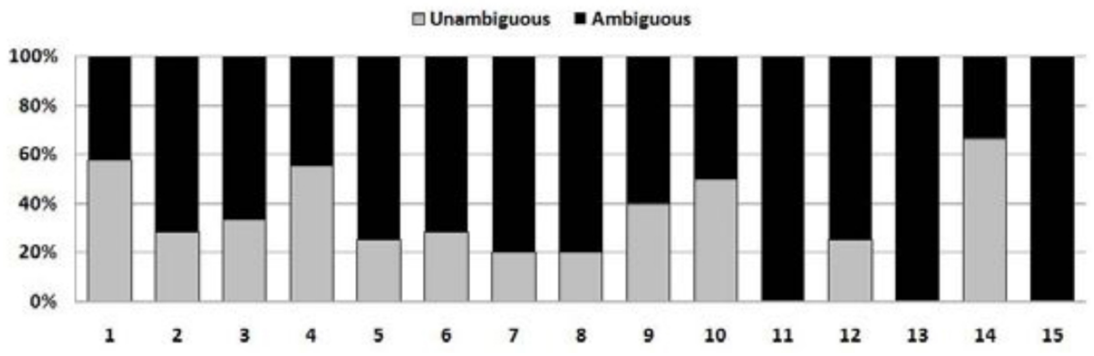


Figure 2.4.: Ambiguity rate in software requirements [Wieringa and Persson, 2010, p.238].

ID	Category	% of total	ID	Category	% of total
1	Functionality A	19,00%	9	Functionality F	4,66%
2	Functionality B	13,98%	10	Usability	4,30%
3	Functionality C	8,96%	11	Security	3,94%
4	Functionality D	8,60%	12	Data model	3,94%
5	Functionality E	7,53%	13	Reliability	3,23%
6	Infrastructure	6,81%	14	Performance	2,51%
7	Maintainability	5,02%	15	Functionality G	2,51%
8	Software features	5,02%			

Table 2.1.: Requirement categories (adapted from [Wieringa and Persson, 2010, p.236]).

System

When we talk about a system in this project a software-based systems is meant. In [Partsch, 2010, p.23] the term *system* is defined as an entity consisting of different components that are cut off its environment after a certain criteria. All subsystems communicate and interact with each other to achieve the same goal. A software-based system is a system that contains at least one software component.

Model

A model in the context of RE, according to [Partsch, 2010], serves as an abstraction of reality. Therefore, the complexity of facts is reduced and the essential features are highlighted. That leads to a better understanding of the different stakeholders and gives a better basis of communication, discussion and development. Thus, models are very important for the requirements engineering process. The challenge, when to use what kind of model is also mastered in this book. The characteristics of models are listed as follows [Partsch, 2010, p.35ff]:

- Mapping: every model has its original, that again can be a model.
- Pragmatic feature: the model is able to replace the original, if some conditions are fulfilled.
- Abbreviation: a model only contains the relevant attributes of the original, but new attributes can be added.

2.1.3. Approach

[IEEE, 1998a, p.15] specifies the development of system requirements as an iterative process, consisting of four sub-processes:

- Identification of requirements: in this step the aggregate set of system requirements is gathered. By identifying the requirements, several parties like customers, the system environment and the technical community need to be considered. It has to be ensured, that the collection is complete and none of the requirements is stated twice. Strategies and techniques to perform this step are listed in [IEEE, 1998a, p.16f].

- Construction of well-formed requirements: the requirements, that are already available as raw statements, are given a well form. This means that they are transformed to a statement of need, that is necessary, short and definite where conditions like qualitative or quantitative measures are defined. Well-formed requirements are easily readable and they contain simple words and phrases, a uniform arrangement and grammatically correct language. They are also testable.
- Organization of requirements: next, a structure is added, according to a comparative definition method. The requirements are grouped by patterns and properties and attributes are defined. In this phase, priorities are added to the requirements that refer to the inputs of customers; also technical approaches are accounted. To perform this organization into ordered sets, different schemes exist. In most cases requirements are assembled into a hierarchy of capabilities in the way that general capabilities are decomposed into subordinate requirements or by the use of network links.
- Representation of requirements: The last sub-process aims at finding the best way to represent the requirements for all the stakeholders that are involved in a project and who need to understand, review, accept or use the SRS. In most cases one single representation form is not enough because the diversity of individuals working with the requirements necessitates the consideration of the different communities. Another reason for combining representation methods is that some representations make it hard to retrieve specific information, show interactions or relate information in different places. Automated tools can help by generating different representations.

The requirements engineering approach is visualized in figure 2.5, where the interaction between sub-processes can be seen. In [IEEE, 1998a, p.19] the following representation methods are listed:

- Textual representation like papers or electronic documents.
- Representation models like physical, symbolic or graphical models or prototypes.

2.2. i*

In this section the i* Framework is described and advantages and new opportunities are listed. The different elements of this framework are enumerated, described and their notation is visualized with images. Practical examples are given to show how to create the different models and how to perform evaluation.

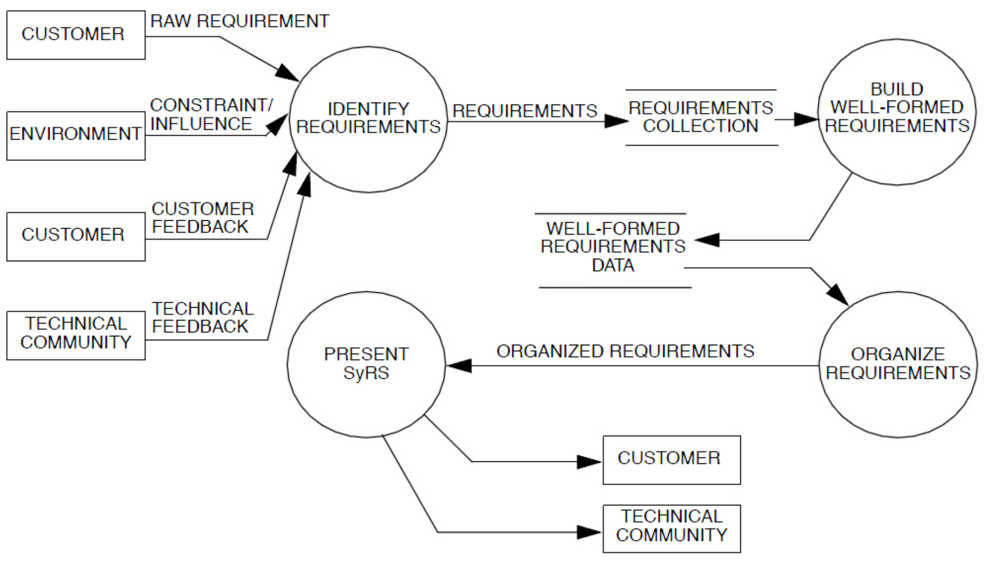


Figure 2.5.: The requirements engineering approach [IEEE, 1998a, p.15].

2.2.1. Introduction to i*

As described in chapter 2.1.2, models are abstractions of the reality that improve the understanding of its original. In software- and information system engineering, techniques that deal with conceptual modeling have been used for many years. These techniques try to find, describe and analyze the implemented behaviors and structures of software in form of static relationships - they are used in entity-relationships models as well as in class diagrams - and dynamic or behavioral properties - they are used in process models as well as in state-based formalisms. The permanent development of new and much more complex software systems in the last decade and their fusion with the social environment of humans, brings a new challenge for developers [Yu, 2009, p.1ff].

An attempt to come to terms with this new challenge is performed with the i* modeling framework, that is proposed and described by Eric Yu in [Yu, 1995]. It brings social understanding into the traditional system engineering process by focusing on social actors. They can be described as active entities in form of humans, hardware, software or a mixture of them, that perform independent actions - with individual goals, beliefs, abilities and commitments. How many goals of a certain actor can be reached under a given relationship-structure is the main task of the analysis phase. The term *i** stands for *distributed intentionality* and it brings intentionality to the context of social networks that consist of autonomous actors [Yu, 2009, p.2ff]. Since 1998 it has become part of an international standard [Yu, 2011].

With the i* Framework also other problems can be solved. In [Liu et al., 2003] a methodological framework for analyzing security and privacy concerns is presented, that is based on agent-based models. With these models, better and richer descriptions and analysis techniques can be used that help to choose between alternatives, discover conflicts and synergies and give a better understanding for implications and consequences. The main purpose of this methodological framework is the identification and handling of security and privacy goals in the very early steps of a project. Therefore the elicitation of security and privacy requirements is integrated into the usual requirements engineering process and is performed together with the addressing of functional and non-functional requirements. With the help of the i* Framework, new ways of analysis can be realized, where for example potential attackers, malicious intentions or vulnerabilities can be found, which leads to a secure software [Liu et al., 2003].

The i* Framework can be used as a graphical representation with different elements, that are combined to build models. Another usage of the i* Framework - beside the graphical one - is the use of the meta-framework, where semantics and constraints of i* are embedded [Mylopoulos et al., 1990], [Liu et al., 2003].

2.2.2. Notation

The i* Framework contains a set of predefined elements that can be used to create different models. This subsection explains the variety of elements and shows the graphical appropriate notations.

Actors

An *actor* is the central construct in the conceptual modeling procedure of i*. It is an active entity that is involved in the system to be created and is able to perform actions independently. Each actor follows individual goals and beliefs [Yu, 2009, p.3f]. To achieve them, it is able to use one's own know-how to carry out actions observing certain restrictions and commitments. Actors are able to intentionally depend on each other. The term *actor* is rather complex and therefore it is a kind of hypernym for three more specialized and concrete elements, that are visualized in figure 2.6 [Horkoff et al., 2006, p.5f]:

- Role: a role is an abstract form of a social actor that contains several characteristics, responsibilities or expectations. Its behavior can be easily transferred to other social actors as well. A role can be played by an agent and all dependencies of the role apply to him as well.

- Agent: an agent is the concrete form of an actor, with physical manifestations. It can be a specific human, machine or software. The agent contains its individual functionalities and capabilities and is able to play one or more roles. Its behavior and characteristics cannot be easily transferred to other social actors like those from a role. The dependencies of the agent result from the roles he is playing.
- Position: a position is a collection of many roles that are assigned to one specific agent. It can be said that the agent occupies the position, that again covers different roles [Liu et al., 2003, p.2ff], [Horkoff et al., 2006, p.5ff].

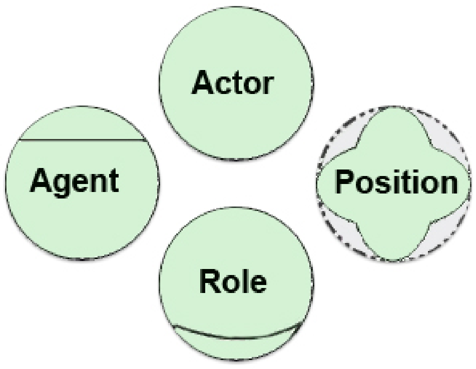


Figure 2.6.: i* notation of actors.

Association Links

Actors can be related among themselves with so-called *association links*. The following six links are defined in [Abdulhadi et al., 2007] and their notation is presented in figure 2.7:

- Is-part-of Association: this relation visualizes actors - roles, positions or agents - that are comprised of subparts in form of other actors. Each actor in such collection is taken to be intentional.
- ISA Association: two actors of the same type can be connected with this association, that describes generalization. Like in some object oriented programming languages, the derived actor is a specialized case of its more general parent.
- Plays Association: it connects an agent to a specific role by showing that the role is played by that agent. The agent's identity and the responsibilities of the role do not effect each other.

- Covers Relationship: with this relation, a position can be connected to several roles. It shows what roles are covered by a certain position.
- Occupies Relationship: to visualize that an agent occupies a specific position - and furthermore that it plays all the roles that it covers - this relationship is used.
- INS Relationship: this relation stands for the instantiation of an actor to a more specific entity of the same type, the instance.

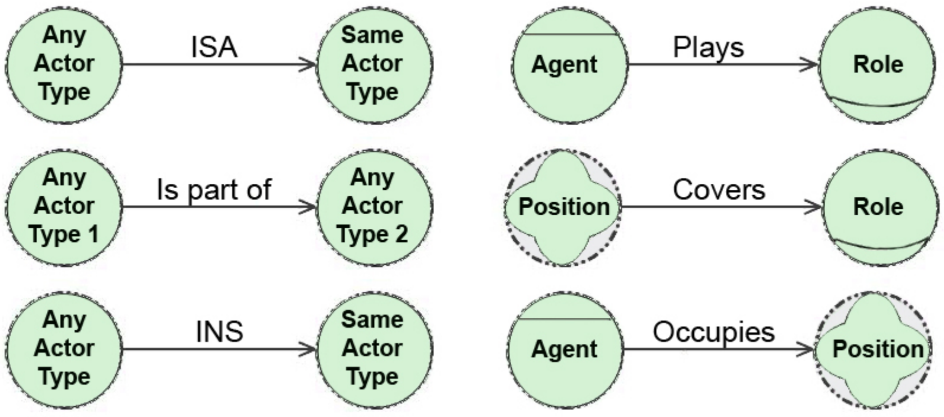


Figure 2.7.: i* notation of the association links.

Strategic Dependency Links

To model the social aspect of i*, *dependency links* are introduced. They describe dependencies among actors regarding the fulfillment of goals, the performance of tasks or the furnishing of a certain resources. A dependency between two actors is called strategic, because it gives the depender - the one who relies on the other actor (who is called dependee) - the possibility to benefit from new opportunities and makes them vulnerable to not receiving what they need. The several types of dependencies are shown in figure 2.8. They are differentiated by the dependum, the central item of the dependency [Yu, 2009, p.1ff], [Abdulhadi et al., 2007]:

- Goal dependency: in this case, the dependum is stated as an assertion. The depender wants the dependee to fulfill this assertion but does not specify how the goal should be achieved. This gives the dependee total freedom in decision making.

- Task dependency: the dependum can also be stated as an activity in a task dependency. Here, the depender wants the dependee to carry out a task with specified activities. Therefore he has already made decisions on how the dependum should be performed, what reduces the freedom of the dependee. Because of the fact, that such specification is usually not complete, the dependee is still able to act on his own with respect to the constraints.
- Resource dependency: if the dependum is a physical or informational object, this is modeled as a resource dependency, where the depender wants the dependee to allocate the certain entity. The entity is provided in the form of a resource that can be consumed by the depender. All the effort that is needed to produce the dependum is brought single handed by the dependee.
- Softgoal dependency: the last dependency centers on a dependum in form of a quality requirement. This dependency is very similar to the goal dependency. The only difference lies in the criteria for achievement which is not exactly defined from the start in the softgoal dependency and further elaboration steps like the consultation between the involved actors are needed.

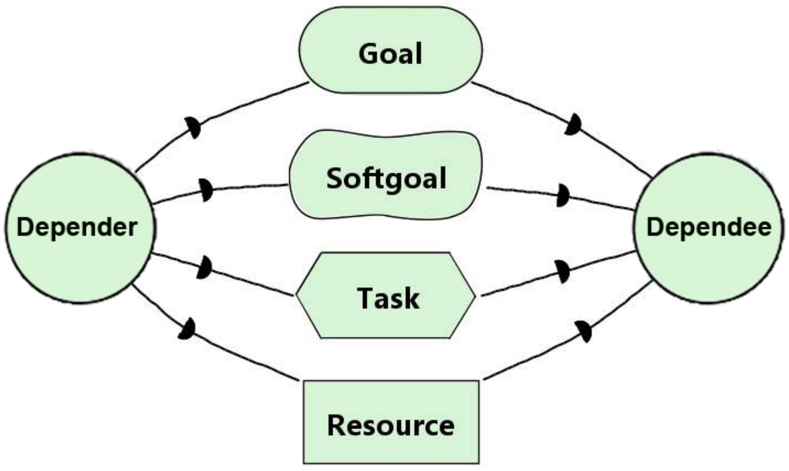


Figure 2.8.: i* notation of the dependency link.

The influence of failing and not providing the dependum by the dependee can be further distinguished. The different levels of dependency strengths describe the degrees of *vulnerability* of the depender [Abdulhadi et al., 2007]:

- Open (Uncommitted) dependency: this is the weakest dependency. If the dependee fails, the depender is influenced only in a small extent. An open dependency is notated with an "O" at both sides of the *dependency link*.

- Open dependency: this is the default dependency and has no additional label. Here, the absence of the dependum causes some depender actions - that are performed to reach a certain goal - to fail.
- Critical dependency: this is the hardest dependency where all actions of the depender - that are performed to reach a certain goal - fail, if the dependum is not obtained by the dependee.

Actor Boundary

Actor boundaries are used to describe the internal makeup of an actor. It includes elements such as goals, tasks or resources and their relations between each other. All elements within the boundary are explicitly desired by a certain actor. Figure 2.9 shows an actor with its boundary in form of dotted line.

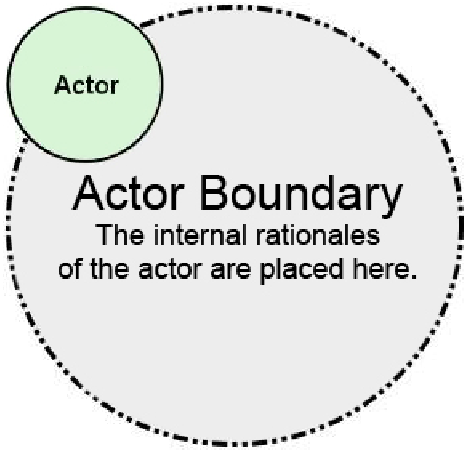


Figure 2.9.: i* notation of the actor boundary.

Elements within an actor's boundary

Within the boundaries of an actor, all the elements that are used as dependum in *dependency links* can be utilized. In contrast to the external dependencies, elements may be accomplished internally. The following elements can be used to describe the actor's internal setup [Abdulhadi et al., 2007], [Liu et al., 2003, 4.ff]:

- Goals: a goal can be separated to a hard- or a softgoal. While the first one describes a function, the second one stands for a quality requirement, where the criteria for the satisfaction of the goal is not entirely clear and it depends on the point of view of the actor. Hardgoals can be achieved differently by tasks that are connected to them with *means-end links*, what is further described later. If at least one task is satisfied, the goal is accomplished. Softgoals instead are described by the *contribution links* from other elements.
- Tasks: a task describes an activity that is performed by agents. It can be divided into subgoals, softgoals, subtasks or resources. This is done through the so-called *decomposition link*. To accomplish a task, all sub-elements need to be accomplished as well.
- Resources: a resource describes the availability of a physical or informational entity, without specifying how this entity is going to be achieved.
- Beliefs: a belief stands for an assumption, domain characteristic or environmental condition that is thought to be true by the actor. Contrary to a goal, the actor does not explicitly want the condition to become satisfied.

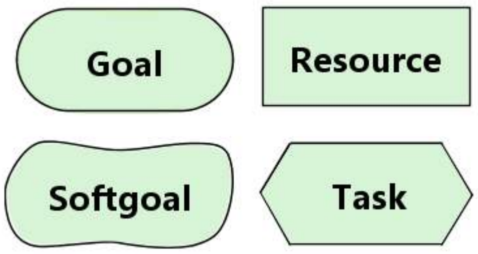


Figure 2.10.: i* notation of additional nodes.

Contribution Links

The influence of any other element on a softgoal can be described with *contribution links*. They define the way of contributing to the achievement of the quality requirement and its strength. This additional information is the most important decision criteria in the analysis phase, where one has to choose between alternative tasks that achieve the same goal. The links And and Or are used to refine a certain softgoal into more specific ones [Yu, 2009, p.6f]. There are nine *contribution links* defined in [Abdulhadi et al., 2007] and in [Chung et al., 2000] as follows:

- Make: make is a positive contribution that provides sufficient positive support. That means, if one offspring is satisficed and connected via this linkage, its parent is satisficed as well.
- Help: it is a positive contribution that only brings partial positive support. A satisficed offspring that is connected via this linkage is not strong enough to satisfice the parent on its own. The parent can only be partially satisficed.
- Some+: this contribution is definite a positive one and it gives positive support. It can be either a make or a help relation, so the strength of the contribution - meaning if the support is sufficient or partial - is not defined. Therefore a satisficed offspring that is connected via this kind of relation can set the parent softgoal either to satisficed or to partially satisficed.
- Break: the break contribution is the contrary of the make contribution. It is a negative relation that provides sufficient negative support. If one offspring is satisficed and connected via this linkage, it gives the parent sufficient negative support so that it gets denied.
- Hurt: hurt is the contrary of the help contribution. It is a negative contribution that provides partial negative support. If a satisficed offspring is connected via this linkage, it is not sufficient enough to deny the parent on its own. The parent can only be partially denied.
- Some-: this relation provides negative support in form of either a hurt or a break contribution. Therefore the strength of the influence is unknown. The connection from a satisfied offspring via this linkage causes the parent softgoal to be denied or partially denied.
- Unknown: such contribution is of unknown sign and of unknown strength. The influence on a parent softgoal is therefore either a positive or negative one with a partial or sufficient extent.
- Or: an or contribution is used to refine a certain softgoal by a group of more specific offspring. If one of them is satisficed, its parent can do so as well.
- And: the last contribution again relates a group of offspring to their parent. If all offspring are satisficed, the parent is satisficed too.

In i* the *contribution links* are sometimes visualized by an arrow containing the proper label, what can be seen in figure 2.11. The arrows point from the offspring to the parent, that is a softgoal [Abdulhadi et al., 2007].

Figure 2.12 illustrates the contribution types that relate a single offspring to a parent, grouped by the polarity of their support.

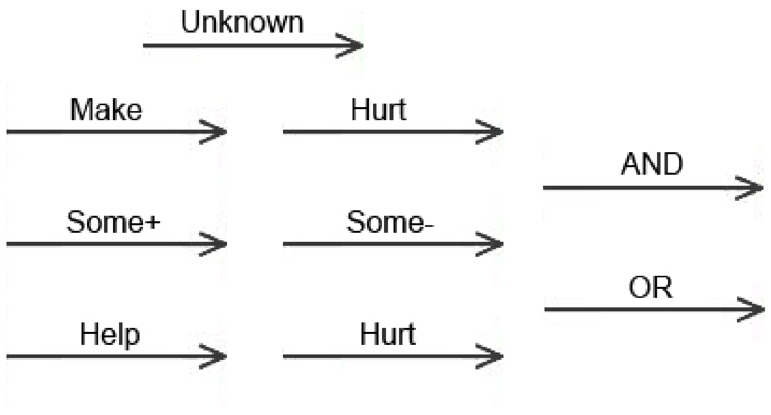


Figure 2.11.: i* notation of the contribution links.

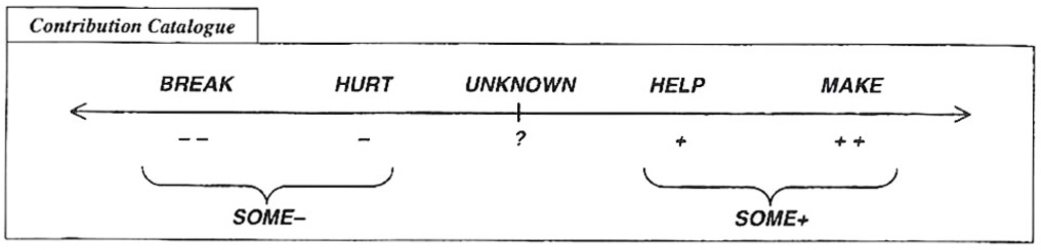


Figure 2.12.: Contribution types grouped by positive and negative support [Chung et al., 2000, p.64].

Decomposition Link

Decomposition links are used to divide a task into its subparts. To illustrate the notation, figure 2.13 shows a simple division of a task in all possible sub-elements that are subgoals, softgoals, subtasks and resources. Each element can appear in an arbitrary quantity. To accomplish the parent task, all sub-elements need to be accomplished as well [Abdulhadi et al., 2007].

- Task - Goal: if the task contains one or more subgoals, alternatives on how to achieve them can be considered. The activities to achieve the goal are not specified.
- Task - Task: a subtask limits its parent to a certain action. It can again be decomposed into its sub-elements.
- Task - Resource: in this decomposition it is important whether the entity is available or not.
- Task - Softgoal: the softgoal represents a quality requirement of the certain task. Such decomposition is useful to choose between alternatives in the further decomposition of a task [Abdulhadi et al., 2007].

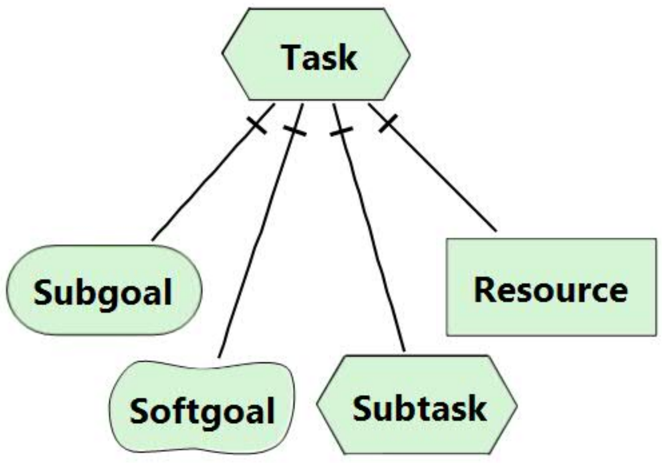


Figure 2.13.: i* notation of the decomposition link.

Means End Link

Such a link combines a task (describing the means) with a goal (describing the end). It shows the ways a goal can be achieved. Figure 2.14 visualizes the graphical notation of that link in form of an arrow pointing from a task to a goal.

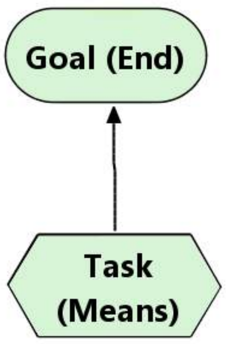


Figure 2.14.: i* notation of the means-end link.

2.2.3. Usage

As mentioned in section 2.2.1 the i* Framework is used to analyze the domain requirements. The analysis steps are defined in [Liu et al., 2003, p.2] as follows:

- Actor identification: in the first step, the actors of the system are identified - *who really is involved in the system?* Every actor can be assigned a role or be further differentiated into agents and positions. Step 1 in figure 2.15 shows this initial step where human actors as well as existing machine actors are identified. After one iteration in the analysis procedure, new actors and system agents are created. This is described by step 5 in figure 2.15. Here, first design choices are made and new functional entities are added.
- Goal/Task identification: now the high-level objectives of agents in form of hard- and softgoals are shaped (step 2 in figure 2.15). This tells *what* different actors want to achieve. Tasks, resources and beliefs are used to describe further characteristics (see section 2.2.1). Step 3 (in figure 2.15) shows the refinement process. Here, hardgoals, softgoals and tasks are sophisticated with *means-end*, *decomposition* and *contribution links*. The refinement is done several times until the model is precise enough to serve as a basis for design decisions.
- Dependency identification: here, the dependency relations between the actors are added to show *how* actors relate to each other (step 4 in figure 2.15). This relations are visualized by *dependency links* (see section 2.2.1), that describe intentional relations rather than information exchange flows.

By performing the analysis steps, different models can be created to deal with different problems. The strategic dependency (SD) model shows a network of actors with its mutual dependencies. It can be created out of an recursive walk through steps 3, 4 and 5 in figure 2.15. The strategic rationale (SR) model is built upon the SD model and internal rationales of the different actors are added. Each goal, softgoal, resource or task dependency gets a delegation relationship across the boundary of every actor [Liu et al., 2003, p.2].

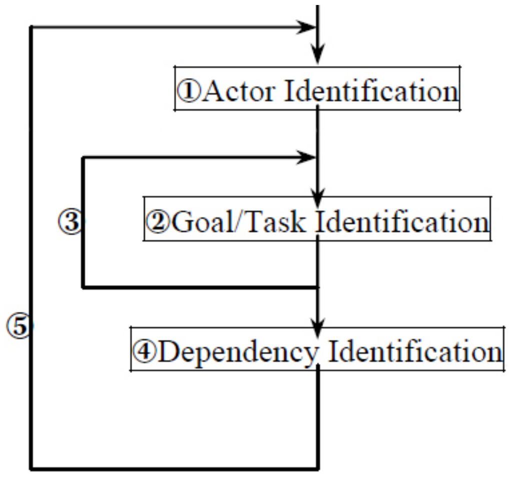


Figure 2.15.: Requirements elicitation process with i* [Liu et al., 2003, p.2].

Strategic Dependency Model

The strategic dependency (SD) model makes use of *dependency links* to construct a network of actors and the intentional dependencies among themselves. Actors can also be connected with *association links* to show how they relate to each other. To define the strength of dependencies, *vulnerabilities* can be used. The analysis of this model gives information about opportunities and vulnerabilities of all the actors involved in the system, that needs to be constructed. An example of such model is provided by figure 3 that shows a simple scenario of a buyer-driven e-commerce system with a middleman. The construction of the SD model needs the recursive execution of steps 3, 4 and 5 that were shown in figure 2.15 [Liu et al., 2003, p.4f], [Yu et al., 2001, p.3f] .

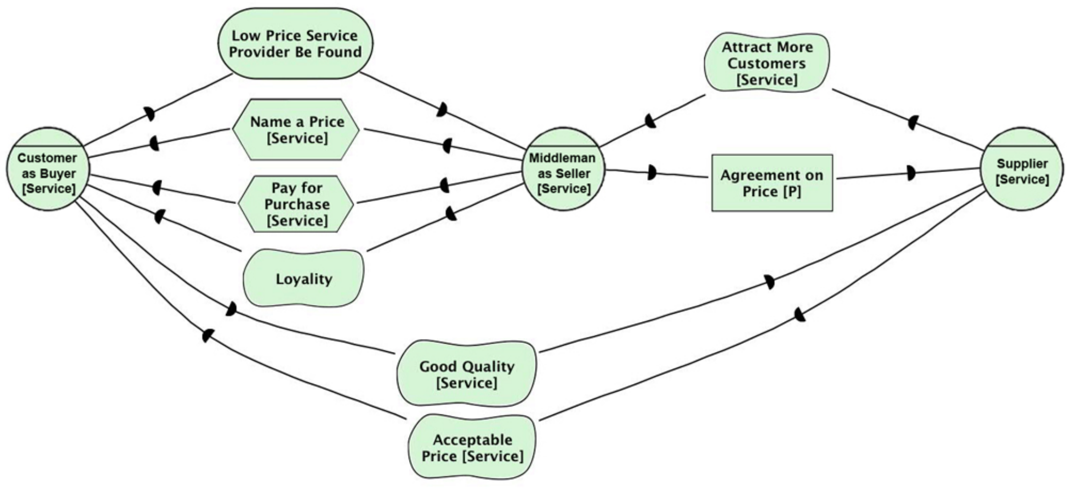

Low Price Service Provider Be Found
Attract More Customers [Service]
Name a Price [Service]
Customer as Buyer [Service]
Middleman as Seller [Service]
Agreement on Price [P]
Supplier [Service]
Pay for Purchase [Service]
Loyality
Good Quality [Service]
Acceptable Price [Service]

Figure 2.16.: SD model example: e-commerce system (adapted from [Yu et al., 2001, p.4]).

Strategic Rationale Model

The strategic rationale (SR) model is an extension of the strategic dependency model and it describes the rationales behind different dependencies. Therefore the actor's internal rationales are added to the existing relations. Each dependency from the SD model is delegated by an internal relationship within the actor boundary. That model makes use of goals, tasks, resources and beliefs. Goals can be achieved by tasks through *means-end links*, tasks can be precisely described with *decomposition links* and the influence of other elements on softgoals can be shown through *contribution links*. Figure 2.17 shows the internal rationales for the buyer-driven e-commerce system with a middleman.

In the analysis phase, the SR model helps to choose between alternative tasks that are able to fulfill the same goal. The decision is influenced by the way they affect the softgoals. To construct the SR model, steps 2, 3 and 4 in figure 2.15 are run through iteratively [Yu, 2009, p.6f], [Liu et al., 2003, p.4f].

i* extensions

[Strohmaier et al., 2007] introduces a method to analyze the effectiveness of knowledge transfer instruments that is based on i*. This method is designed to facilitate the knowledge transfer between knowledge workers which can be seen as one of the main challenges in knowledge management. Some approaches to come to terms with this goal have already existed in form of so-called "knowledge transfer instruments". They analyze the problem from different points of view [Strohmaier et al., 2007, p.1]:

→ technological: the technological approach can be realized by knowledge management systems or knowledge infrastructures.

→ organizational: on organizational level, the "Experience Factory" concept helps to improve the knowledge transfer.

→ social: examples are communities of practice.

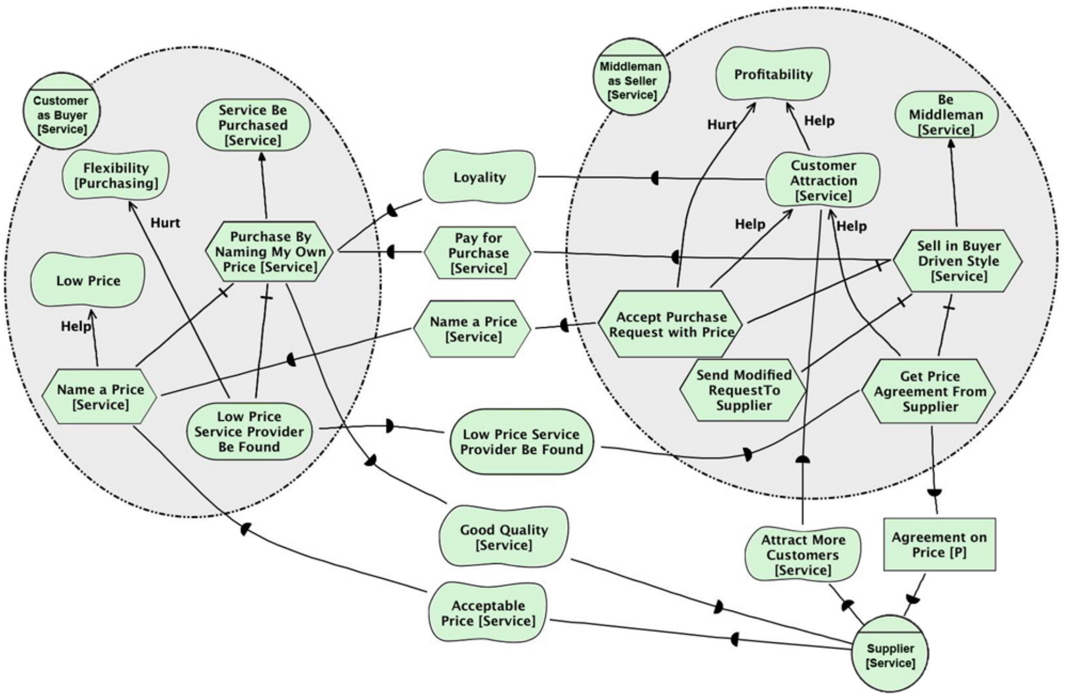

Customer as Buyer [Service]
Service Be Purchased [Service]
Flexibility [Purchasing]
Hurt
Purchase By Naming My Own Price [Service]
Low Price
Help
Name a Price [Service]
Low Price Service Provider Be Found
Loyality
Pay for Purchase [Service]
Name a Price [Service]
Low Price Service Provider Be Found
Good Quality [Service]
Acceptable Price [Service]
Middleman as Seller [Service]
Profitability
Hurt
Help
Be Middleman [Service]
Customer Attraction [Service]
Help
Help
Sell in Buyer Driven Style [Service]
Accept Purchase Request with Price
Send Modified RequestTo Supplier
Get Price Agreement From Supplier
Attract More Customers [Service]
Agreement on Price [P]
Supplier [Service]

Figure 2.17.: SR model example: e-commerce system (adapted from [Yu et al., 2001, p.5]).

Many studies showed the importance to regard stakeholders and their goals by analyzing the knowledge transfer effectiveness. Their motivation and acceptance also play an important role in the quality criteria. The traditional approaches pay less attention to modeling the goals and intentions of knowledge transfer participants and instruments. Therefore the new Knowledge Transfer Agent (KTA) Modeling Method is introduced as an extension of the i* Framework. It is split up into three phases with different levels of detail and analysis possibilities. This opens the possibility to choose the phase that fits best to the requirements of a certain situation [Strohmaier et al., 2007, p.12f].

Level 1 - Identification of knowledge dependencies: Here, the strategic knowledge dependencies of actors are identified in form of *dependency links*. The dependum in form of knowledge is an extension of the i* Framework that helps to identify knowledge risks in organizations or knowledge networks and communities.

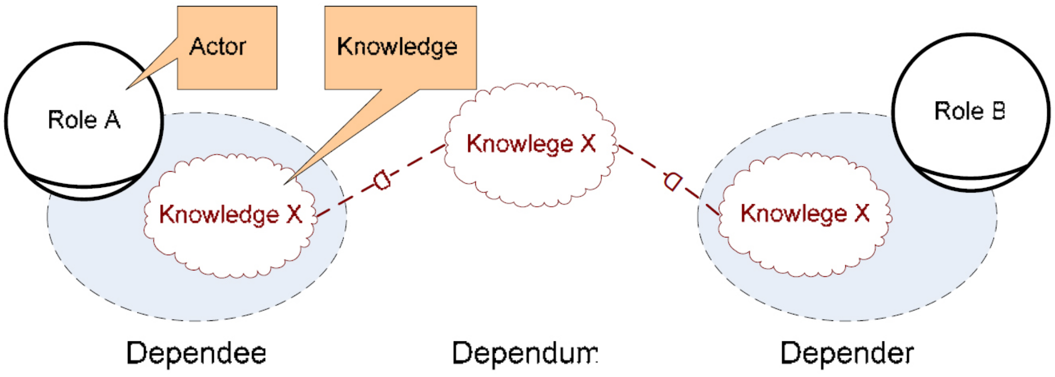


Figure 2.18.: Identification of knowledge dependencies [Strohmaier et al., 2007, p.4].

Level 2 - Identification of supportive means per dependency: This level describes how the knowledge transfer takes place by adding communication channels and storage objects. The first item is used for intentional knowledge transfer from sender to receiver like a face-to-face communication or phone call. The second item is used to store information and make it usable for other actors. Both elements can be connected to a knowledge dependency through *means-end links*. To assign different means to the same end helps to find alternative knowledge transfer instruments.

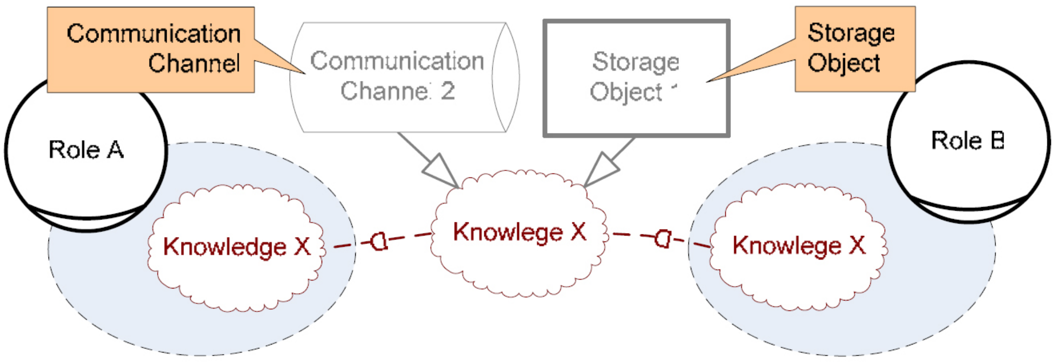


Figure 2.19.: Identification of supportive means [Strohmaier et al., 2007, p.5].

Level 3 - Re-conceptualization of supportive means: In the highest level of detail, the knowledge dependencies and its supportive means are transformed to an individual agent, the Knowledge Transfer Agent. This represents the own pursuit of goals and the possibility for dependencies that this element has in practice. The re-conceptualization opens the possibility to analyze the goal achievement of knowledge transfer instruments and revokes the need of the additional elements, that were introduced in previous steps.

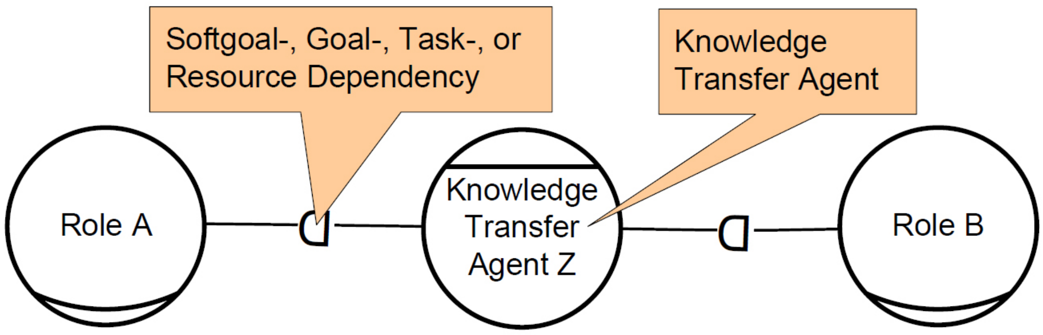

Softgoal-, Goal-, Task-, or Resource Dependency
Knowledge Transfer Agent
Role A
Knowledge Transfer Agent Z
Role B

Figure 2.20.: Re-conceptualization of supportive means [Strohmaier et al., 2007, p.5].

[Strohmaier et al., 2007] also illustrates how the Knowledge Transfer Agent Modeling Method can be combined with existing concepts by applying the three procedures on the "Experience Factory", an approach that aims at the facilitation of knowledge transfer between software developers. It is demonstrated, that the advanced modeling methods help to answer additional questions, that could not be addressed before.

Goal Evaluation

The way goals are evaluated with i* can be formalized by an algorithm. [Horkoff and Yu, 2009a] describe such an algorithm that assigns qualitative evaluation labels to elements of the i* models. The defined labels are satisficed, partially satisficed, conflicted, unknown, partially denied and denied. Two steps are performed by this algorithm:

- Assignment: this is the initial step, where one or more evaluation labels are assigned to elements. The decision on performing this step considers an interesting domain question.

- Propagation: in the next step, the propagation of the first labels are constructed over the whole model. To perform the propagation, several guidelines exist. [Chung et al., 2000] illustrates this process in two procedures. First, the "individual impact" of an offspring to its parent according to the specific *contribution link* is determined. After that, parents with more than one offspring are handled. This is done by combining all the individual impacts into a single label, where sometimes judgments of the domain expert are needed. Figure 2.21 shows the individual impacts of an offspring to its parent under different circumstances, that are [Chung et al., 2000, p.72f]:

  – Make: this relation propagates satisficed and denied from the offspring to the parent.

  – Help: the help relation weakens satisficed to partially satisficed and denied to partially denied.

  – Some+: here, the parent receives a weaker label with the same direction as the offspring. This means that satisficed or partially satisficed becomes partially satisficed and denied or partially denied becomes partially denied.

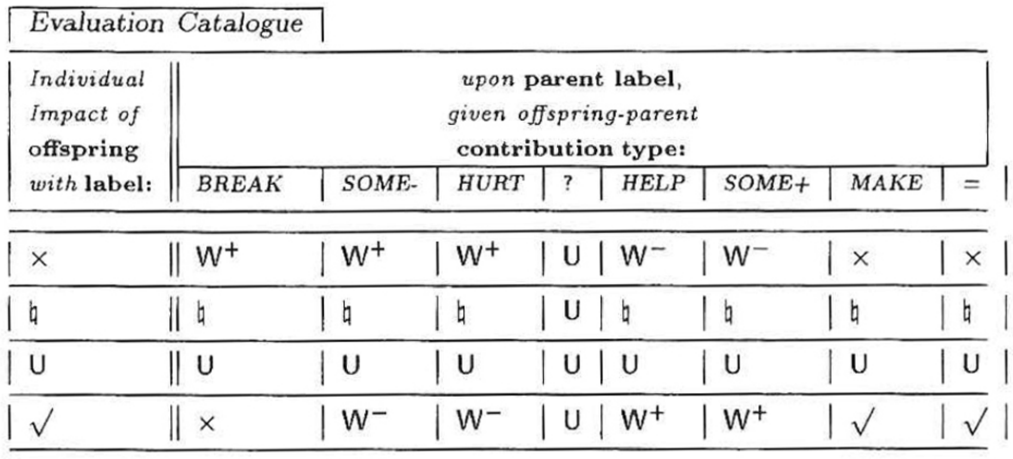
  – Break: break propagates the inverted label as the make contribution, where satisficed goes to denied and denied goes to satisficed.

  – Hurt: it is the inverted help contribution. Satisficed becomes partially denied and denied becomes partially satisficed.

  – Some-: the parent gets the inverted label of the some+ contribution. Satisficed or partially satisficed result in partially denied and denied or partially denied result in partially satisficed.

  – Unknown: the unknown contribution propagates unknown to the parent.

  The combination of individual impacts to a single label is done by choosing the minimal label out of a strict order: conflicted $\leq$ unknown $\leq$ denied $\approx$ satisficed. Partially satisficed and partially denied need to be transformed by the developers into one of the four labels, according its direction. Expertise with non functional requirements and knowledge of the domain and development provide help for this choice. Sometimes the given situation makes it hard for developers to come to an obvious decision. In that case they have the possibility to extent the rules by changing the resulting parent to one of the weaker labels (partially satisficed or partially denied). This results in handling the resolution during subsequent evaluation as input type. Therefore the rules of the label satisficed and denied of figure 2.21 could be used by weakening the resulted value [Chung et al., 2000, p.76ff].



Evaluation Catalogue

Individual Impact of **offspring** *with* **label:**	*upon* **parent label,** *given offspring-parent* **contribution type:**							
	BREAK	*SOME-*	*HURT*	*?*	*HELP*	*SOME+*	*MAKE*	*=*
×	W^+	W^+	W^+	U	W^-	W^-	×	×
♮	♮	♮	♮	U	♮	♮	♮	♮
U	U	U	U	U	U	U	U	U
√	×	W^-	W^-	U	W^+	W^+	√	√

Figure 2.21.: Individual impacts in the goal evaluation process [Chung et al., 2000, p.74].

3. Requirements Analysis and System Design

This chapter describes how the requirements are elicited and sorted out in this project to create a basis for the design of the system. Furthermore, it shows how it is possible to thereby satisfy the different stakeholders. Examples and figures state how the requirements are mapped to the different i* models and how they are used and analyzed to find the best fitting technology for the project realization. Another focus of this chapter is to state arguments and justifications for the chosen technology.

3.1. Requirements Elicitation

3.1.1. Actor Identification

In the first weeks, several meetings and talks with employees and different stakeholders are performed to help on understanding the statements of the problem and to give a theoretical introduction to the term *knowledge management*. With the information that is acquired in the meetings, the strategic dependency model - that is described in chapter 2.2.3 - is created according the defined steps.

Initially, the active entities that are involved in this project are identified in form of agents and roles. The involved parties can be humans, hardware, software or a combination of them. The result of this first step is shown in figure 3.1. The following actors are determined:

- *Passive User*: the *Passive User* agent is an operator of the software, the one with the most restricted access. He just utilizes the *Management Tool* to request information. Therefore he performs search queries to find needed data and watches the project progress. This kind of actor does not influence the existing data but only reads out the needed project specific information. Because of the inter-site collaboration within the company, the software must offer the possibility for inter-site search and retrieval. The passive

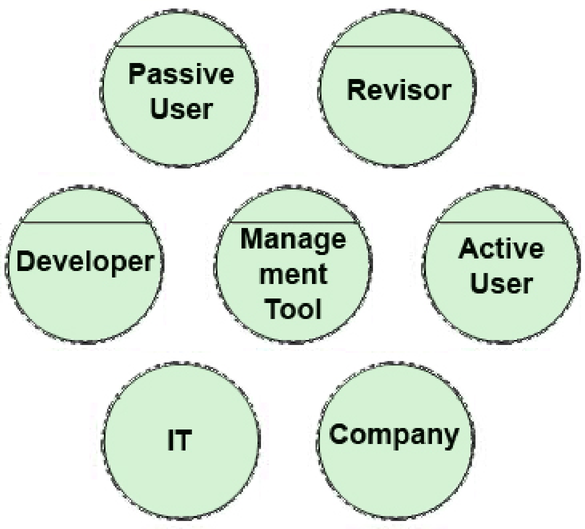


Figure 3.1.: Identified stakeholders of the software system.

user needs a simple and usable interface for his work that represents the requested information in a clear way. Despite the huge amount of data, he does not ask for long waiting times for his requests.

- *Active User*: the *Active User*, that is modeled as an i* agent - is able to maintain the *Management Tool* in a certain way. He makes changes to the information content by adding and adapting data. This actor is responsible for keeping the project data up to date and for continuously updating the project progress by assigning files to milestones. So his know-how is of importance. To perform gapless records the data management must be usable and unified and a proper user interface needs to be provided. The *Active User* also requires a simple and efficient storage for project specific documents and the possibility for inter-site knowledge deployment.
- *Revisor*: this user has the rights to review documents that are assigned to the system and thereby contributes to the correctness of the information content. The reviewed data must be identified properly. The possibility to label documents helps this actor to monitor the progress of a test project, to identify missing files for a milestone and to advise the project owner in that case.
- *Developer*: the *Developer* is added as stakeholder for the sake of completeness. He also plays a role in the dependency network because he chooses the technology and brings his know-how to the realization process. This agent wants to restrict the effort to an assessable one and desires freedom according

the environment in pursuing his target, the implementation and preparation of a useful and accessible software-based system.

- *IT*: this actor represents the IT department of the company. It provides the needed environment capabilities and is able to define security standards and other restrictions for the development. The *Management Tool* needs to consider all the rules that are issued by this party like the prohibition to use software that needs a license. The software must be realized in a way that minimizes the maintenance effort for the IT department, that is responsible for keeping internal software in good condition.
- *Company*: the *Company* as actor is interested in the accomplishment of this project, because internal processes are improved. Furthermore this update bring savings in money and time. The company provides the funds for the project.
- *Management Tool*: this agent is the central element of the project and represents the management tool that is introduced to the company in form of a software-based stakeholder. The application needs to be implemented and prepared in a way that takes account of all requirements from the different stakeholders. The more demands are fulfilled, the merrier it is. A proper technology is chosen that fits best in this specific statement of the problem and gives the freedom to squeeze the software to the claimed shape. This actor is implemented by the developer and accepted and maintained by the *IT*.

As the main actors are determined, the relation-network is created to represent the concrete situation of the company. It shows what rights are allocated to the different internal divisions and how they are able to use the software. Therefore some new actors need to be added. To set up the connections between the actors, *association links* are used. This step is visualized in figure 3.2.

The connection between the concrete departments of the company and the agents for the three user types that were identified in the last step is modeled with roles. For each type, a proper role is added, that respectively depends on a concrete user agent.

- *Active SW User*: this role is played by actors, that operate as active software users. The role depends on the agent *Active User* to offer the resource of a concrete user.
- *Passive SW User*: it describes the role that is played by actors performing the work of a passive software user. A dependency exists to the concrete *Passive User* agent.

- *SW Admin*: the role of the software administrator is only played by those agents, that are privileged to perform reviews of existing content. It depends on the concrete agent *Revisor*.

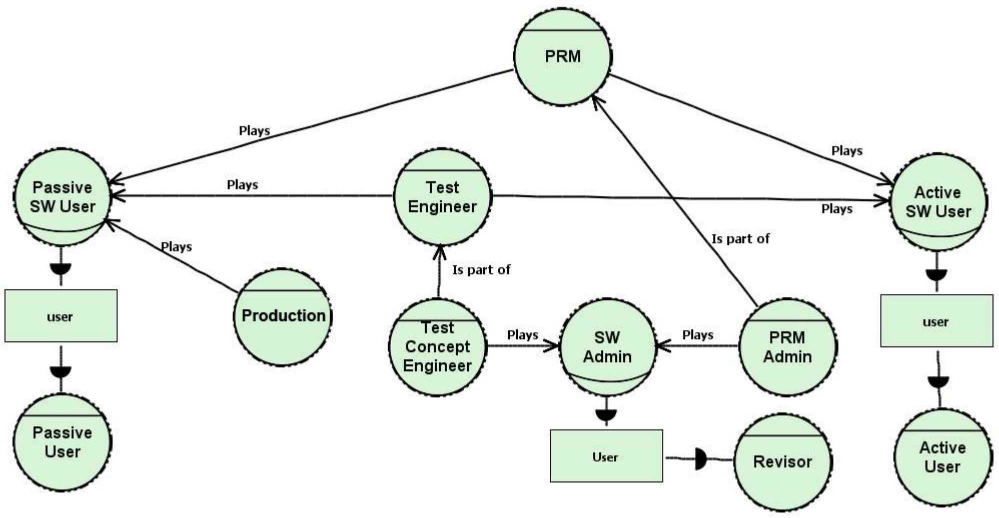


Figure 3.2.: Stakeholders, related with association links.

There are three main departments of the company involved in this project:

- *Production*: this agent represents an employee of the production department. Such actor only behaves as a *Passive User* that is restricted to read-only access. That is because the production is not directly involved to the test engineering processes and does not always bring the needed know-how to add or change test documentations. Nevertheless, there are many situations in which information about the test engineering process is needed and the tool is used to allocate it.
- *Test Engineer*: with this agent an employee of the test development department is modeled. This actor brings the appropriate know-how to keep the information content of the system up to date because he is directly involved in certain projects. On the other hand he also needs to watch the project progress and to request certain documents via the search engine. Therefore he can play the role of an *Active User* as well as those of a *Passive User*.
- *PRM*: according to the usage of the new software, the so-called product managers are similar stated as *Test Engineers* because they also appear both as *Active* and *Passive User*.

The supervisors of two departments are stated as separate elements, because they are equipped with different permissions:

- *Test Concept Engineer*: this agent is a *Test Engineer* with additional permissions. He acts like a supervisor for the *Test Engineers* and is able to play all three possible roles. To model the inheritance, the ISA relation is used. A *Concept Engineer* has control functionality and is responsible for reviewing milestone specific files. He is also interested in monitoring the project progress and reacting on blockades in form of missing or wrong information.
- *PRM Admin*: the *PRM Admin* is the supervisor of the *PRM* and is similar to the *Test Concept Engineer*. He also plays all three roles and follows related same interests.

3.1.2. Strategic Dependencies

After relating the actors' elements in form of goals, tasks and resources are added according to the rules of the strategic dependency model. They help to visualize the stakeholder's suggestions by defining dependencies among actors. In this case mainly dependencies from and to the *Management Tool* exist, because this actor is the central element of the project. The final SD model is shown in figure 3.3.

Because of the fact, that the produced models are very complex and a detailed description would blast the scope of this work, we only focus on the sub-model of the *Active User*, which is marked with the red square in the SD model (figure 3.3). The sub-model is shown in figure 3.4.

The *Active User* depends on the software that is created through six softgoals, that can be described as quality requirement, where the criterion of achievement is not exactly defined from the beginning.

- *maintainability be provided*: the customized document management system needs to be maintainable for the *Active User*, who is responsible for its content. He wants to add new data and update the existing one. Another point that can be assigned to this softgoal is the possibility to add new projects to the system and indicate it to the search engine.
- *unified document management be provided*: there should only be one central tool to manage all relevant documents. Instead of using emails, shared directories or other tools, the documents should be managed only by the new software. This avoids problems due to different versions of specific test documents.

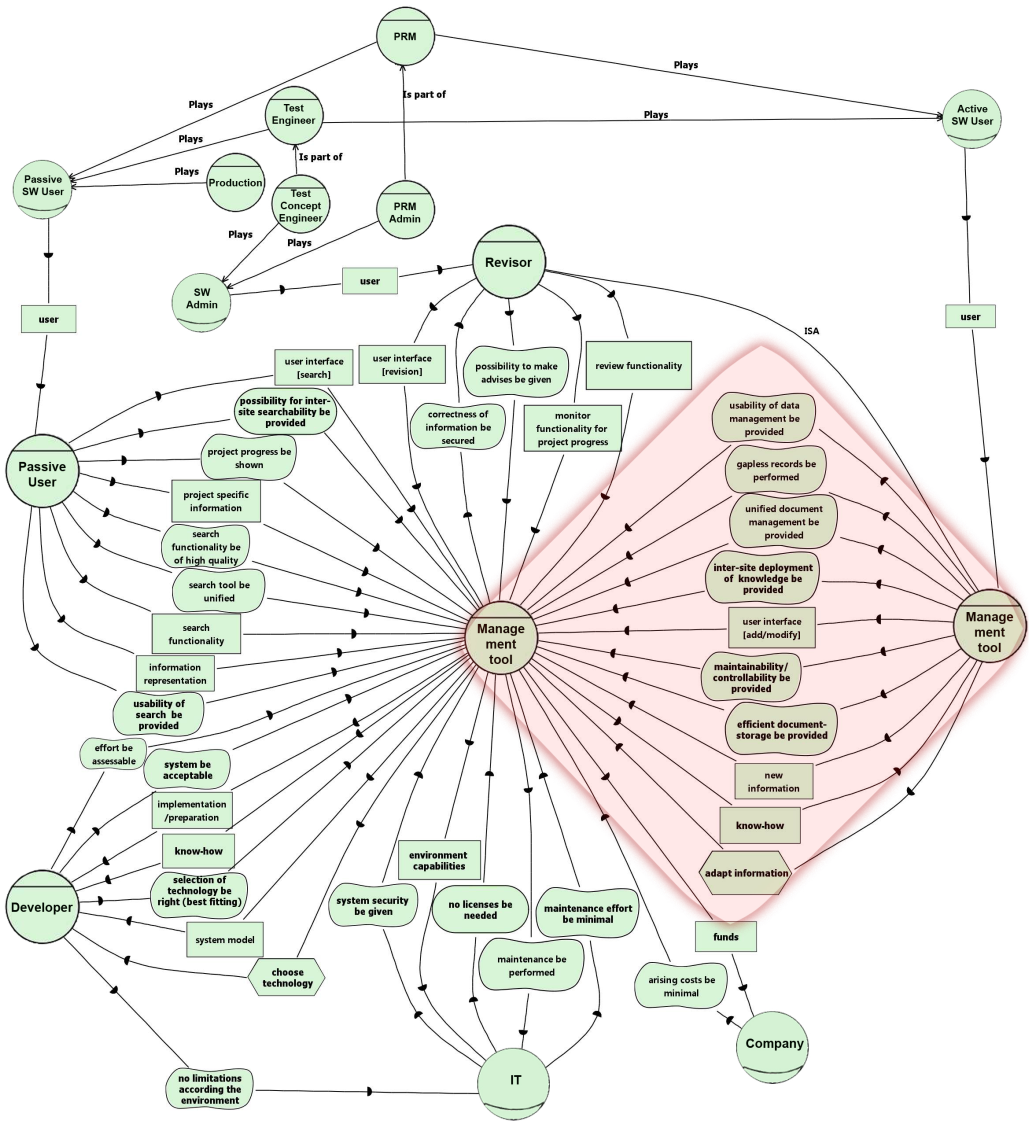

PRM
Plays
Is part of
Test Engineer
Plays
Active SW User
Plays
Passive SW User
Plays
Production
Is part of
Test Concept Engineer
PRM Admin
Plays
Plays
SW Admin
user
Revisor
user
user
ISA
user interface [search]
user interface [revision]
possibility to make advises be given
review functionality
possibility for inter-site searchability be provided
correctness of information be secured
monitor functionality for project progress
usability of data management be provided
Passive User
project progress be shown
gapless records be performed
project specific information
unified document management be provided
search functionality be of high quality
inter-site deployment of knowledge be provided
search tool be unified
user interface [add/modify]
Manage ment tool
Manage ment tool
search functionality
information representation
maintainability/ controllability be provided
usability of search be provided
efficient document-storage be provided
effort be assessable
system be acceptable
new information
implementation /preparation
know-how
know-how
environment capabilities
adapt information
selection of technology be right (best fitting)
Developer
system security be given
no licenses be needed
maintenance effort be minimal
funds
system model
maintenance be performed
choose technology
arising costs be minimal
Company
no limitations according the environment
IT

Figure 3.3.: The strategic dependency model.

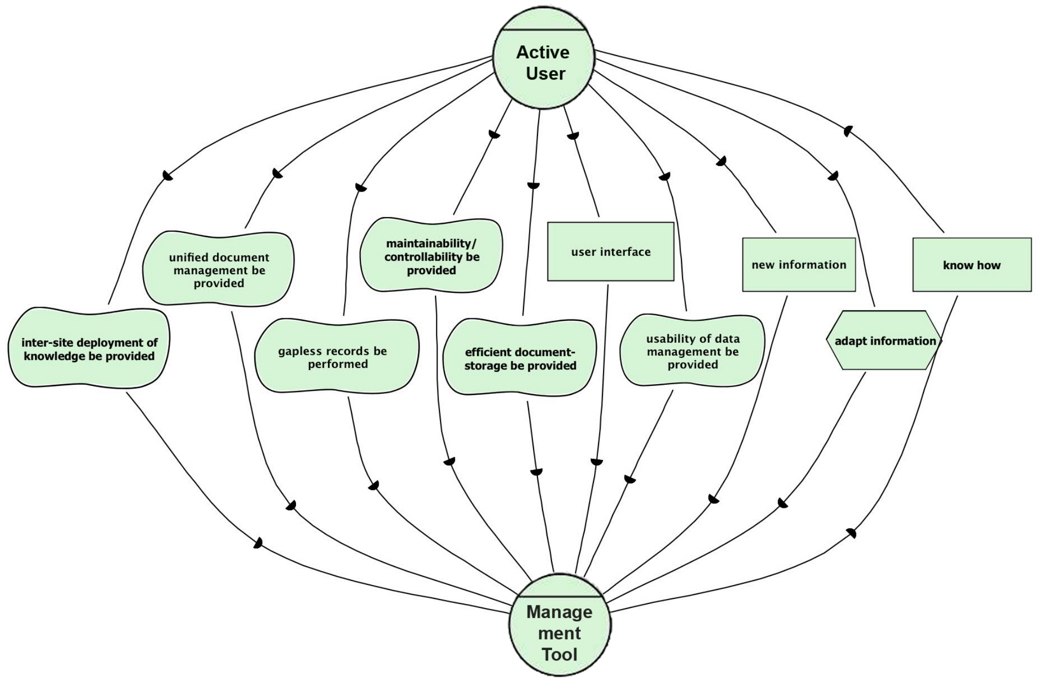

Active User
unified document management be provided
maintainability/ controllability be provided
user interface
new information
know how
inter-site deployment of knowledge be provided
gapless records be performed
efficient document-storage be provided
usability of data management be provided
adapt information
Manage ment Tool

Figure 3.4.: Strategic dependency sub-model.

- *possibility for inter-site knowledge deployment be provided*: products are often treated in different sites and in different countries. To ease the information transfer, the new software should provide a possibility for inter-site knowledge deployment.
- *gapless records be performed*: the *Active User* needs to perform gapless records to keep the content up to date. Therefore, it is important that performing such records is simple, can be done quickly and furthermore is accepted by every active user.
- *usability of data management be provided*: the usability of the data management is a very important point for the *Active User*. No complex processes are allowed to be introduced that need further trainings or lead the operators to make mistakes. Usability helps to ensure the utilization of the new software.
- *efficient document storage be provided*: documents containing project information need to be stored in an efficient way where everyone can find them. It should be possible to open or adapt a file in a common way.

Moreover, the *Active User* depends on the *Management Tool* that provides two resources. In these cases he wants the software to provide or allocate physical or informational objects:

- *user interface [add/modify]*: the software needs to provide a user interface that allows adding new data to the system and modifying or removing existing data. Also the assignment of data to milestones is performed by this interface.
- *new information*: the *Active User* wants the system to provide and visualize information that has been added to the system.

To be able to adapt or modify the existing information of the system, the *Active User* depends on the new software upon the task dependency *adapt information*. In that case, the object of the dependency is an activity.

On the other side, the new software depends on the *Active User* who provides the softgoal *information be kept up to date* and the resource *know-how*. While the first one signifies that the system can only be used properly if the information is kept up to date, the second one shows that knowledge of the *Active User* is required to create it.

The current model helps to structure and furthermore to understand *what* requirements need to be considered by creating the new software and *where* they come from. The different stakeholders and their needs are visualized in this model. Because of the fact that such figure can be easily understood and that it gives a broad overview of all factors of influence, it can be perfectly used in meetings as conference criteria.

3.1.3. Strategic Rationale

Based on the SD model, the internal relations are added to show the reasons for dependencies and the influence of softgoals. This is done by appending and relating internal goals, tasks, resources and beliefs according to the rules of the strategic rationale model. The additional information brings much more details to the model and it describes *why* the requirements exist. The analysis of this model can help to make decisions, like in our case, the choice between different technologies to implement this information system. The internal relations of the software tool are not modeled at this point because first it has to be clarified, what the potential technologies are.

An overview of the SR model is given by figure 3.5. The complexity of this model is much higher than the one of the SD model. Therefore the figure serves as an overview and again we only focus on the sub-model showing the *Active User*. It is marked with a red square in the strategic rationale model. The sub-model is presented in figure 3.6.

External dependencies of the *Active User* to the *Management Tool* and the other way around remained the same in the SR model. The circle-icon of the *Active User* is opened to see the internal boundary, where new elements are added.

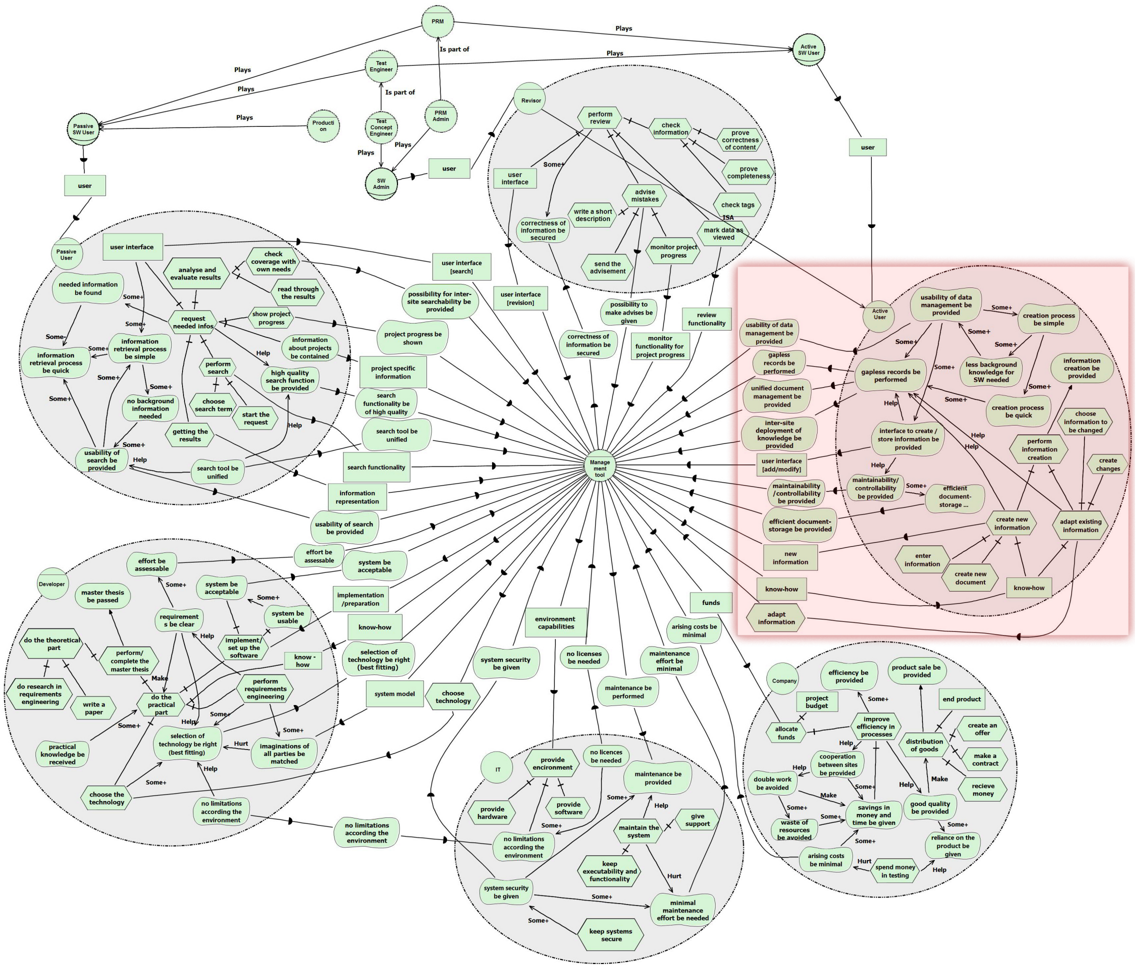


Figure 3.5.: Strategic rationale model.

The goal *information creation be provided* of the *Active User* represents the need to create information to the system. This goal can be achieved by the task *perform information creation*, that is split into the following subtasks:

- *create new information* is one task to *perform information creation*. It describes the process of bringing new information to the *Management Tool* that consists of *create new documents* and *enter information*. Creating new information depends on the *Management Tool* upon the allocation and visualization of information that has been added to the system.

- *adapt existing information* is one task to attain the goal *perform information creation*. It is part of the task dependency between the *Active User* and the *Management Tool* and can be further divided into *choose information to be changed* and *create changes*.

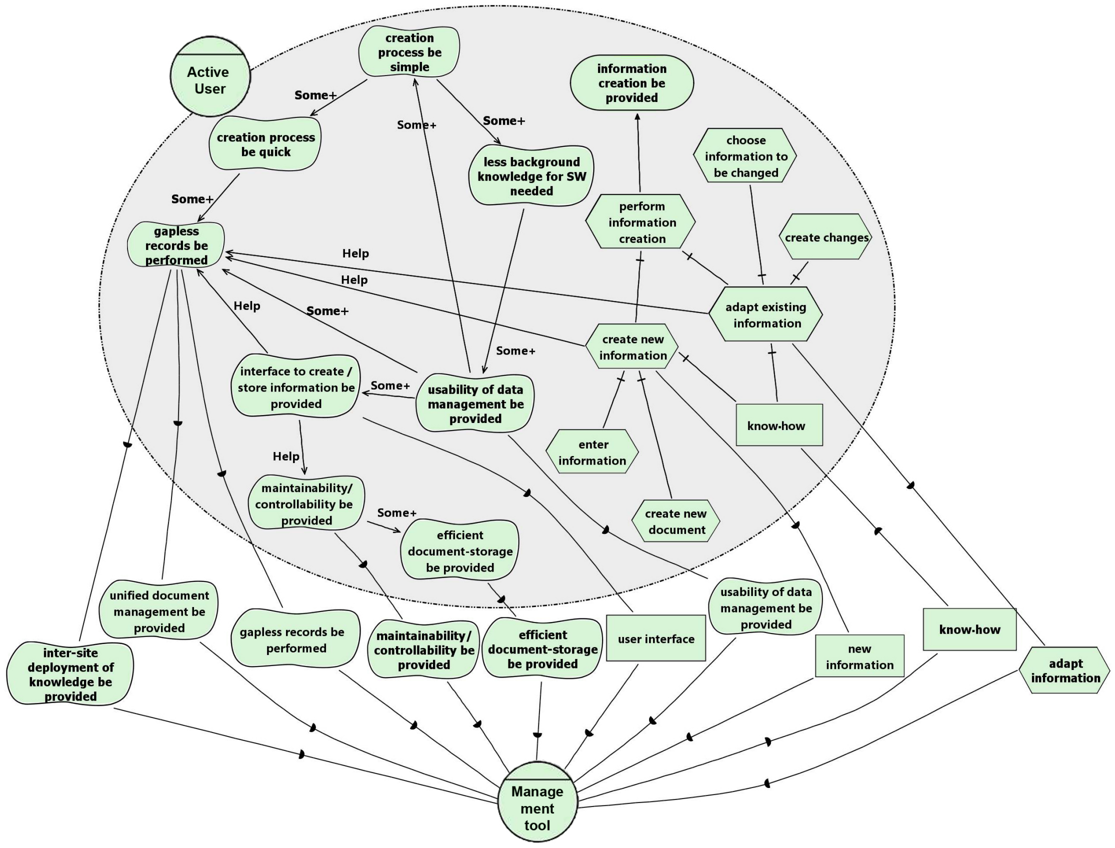

Active User
creation process be simple
information creation be provided
Some+
creation process be quick
Some+
Some+
less background knowledge for SW needed
choose information to be changed
Some+
perform information creation
create changes
gapless records be performed
Help
Help
adapt existing information
Help
Some+
Some+
create new information
interface to create / store information be provided
Some+
usability of data management be provided
know-how
Help
enter information
maintainability/ controllability be provided
Some+
efficient document-storage be provided
create new document
unified document management be provided
usability of data management be provided
gapless records be performed
maintainability/ controllability be provided
efficient document-storage be provided
user interface
know-how
inter-site deployment of knowledge be provided
new information
adapt information
Manage ment tool

Figure 3.6.: Strategic rationale sub-model.

The resource *know-how* is part of the *dependency link* from the *Management Tool* to the *Active User*. It can be associated with both parent tasks and is modeled as a sub-resource. This illustrates, that the *Active User* needs a certain knowledge to create new information and to adapt existing one.

A lot of softgoals exist within the boundary of the *Active User*:

- *efficient document-storage be provided* represents the softgoal that acts like a dependum between the *Active User* and the *Management Tool*. This means that it is part of the *dependency link* between the two actors. To be able to model internal rationales, this element is stated twice, as external dependency as well as internal rationale. This softgoal signifies that the user requires an efficient storage for his project documents.

- *maintainability/controllability be provided* represents the internal rationale of the *dependency link* to the *Management Tool*. The *Active User* requires an easy possibility for maintaining the system and its search index. This softgoal has a positive influence on the first one, what is shown by the some+ relation.
- The softgoal *interface to create/store information be provided* indicates that the information creation process can only be performed if an interface is provided by the *Management Tool*. This interface must also offer functionality to store information. Its provision saves time and resources and builds the basis to make the software usable, what furthermore helps to perform gapless records. Another positive effect is given by this softgoal in form of an increased maintainability and controllability of the management tool.
- *gapless records be performed* is part of three dependencies between the user and the software. This internal softgoal wants the *Management Tool* to provide functionality for knowledge deployment within different sites, a unified document management and the functionality to easily perform gapless records. All of these dependencies come together to make continuously records and updates of information possible.
- *creation process be quick* stands for the disinterest of the *Active User* to perform information creation if it takes a lot of time. A quick creation process has a positive effect on *performing gapless records*.
- *creation process be simple* signifies that this actor is not willing to perform complex processes just to create information within the system. This softgoal has a positive influence on two other softgoals, namely *creation process be quick* and *less background knowledge for the software needed*.
- *less background knowledge for the software needed* shows that the user wants to be able to use the software on its own. He does not want to be addicted to additional information or courses just to create information within the system. If only few background knowledge is needed, the *usability of the data management* is increased.
- *usability of data management be provided* represents the desire to a usable data management. This softgoal is part of a *dependency link* between the *Active User* and the *Management Tool* and it influences the softgoals *creation process be simple*, *gapless records be performed* as well as *interface to create/store information be provided* in a positive way.

3.2. Technology Choice

One big challenge in creating a software-based system is the technology decision. As the requirements are elicited, the amount of possible technologies are in fact restricted, but in most cases there is still more than one way to implement and realize a certain system.

3.2.1. Potential Technologies

In this project after clarifying and modeling the stakeholder's needs, at least two possible implementation types remain:

- Traditional Document Management System

  The requirements could be satisfied by a traditional document management system or short DMS. In that case only one software element exists that is used for adding and modifying information. Also the storage and representation of files is performed here with the help of an internal database. Without that kind of software, the data cannot be touched by users. Additional functionality in form of comments, search functionality, right management, etc. can also be used by a document management system.

  The company already works with two types of document management systems, the so-called "Documentum" and a similar tool called "TPac". Both system are completely integrated and maintained by the IT sector, so no further requests and checks need to be performed. If one of these two DMS could be adapted according to the stakeholder's needs it would save a lot of time, work and problems.

- Search Engine

  Another approach to fulfill the requirements is to implement a kind of search engine in form of an intranet page. The search engine contains a search index, where only metadata of the files are stored. Another tool that fills the search index with data is needed as well. Such application can be implemented in form of a file crawler that walks periodically through the file system and indicates metadata or in form of a file monitor, that observes the file system and indicates changes. A data storage is provided separately in form of network shares.

  By constructing and introducing a new software-based system to the company, several guidelines and restrictions must be considered. The basic requirement determines that the software must be easily maintainable and independently runnable without further help for a long time period. The system also needs to be accepted by the IT department.

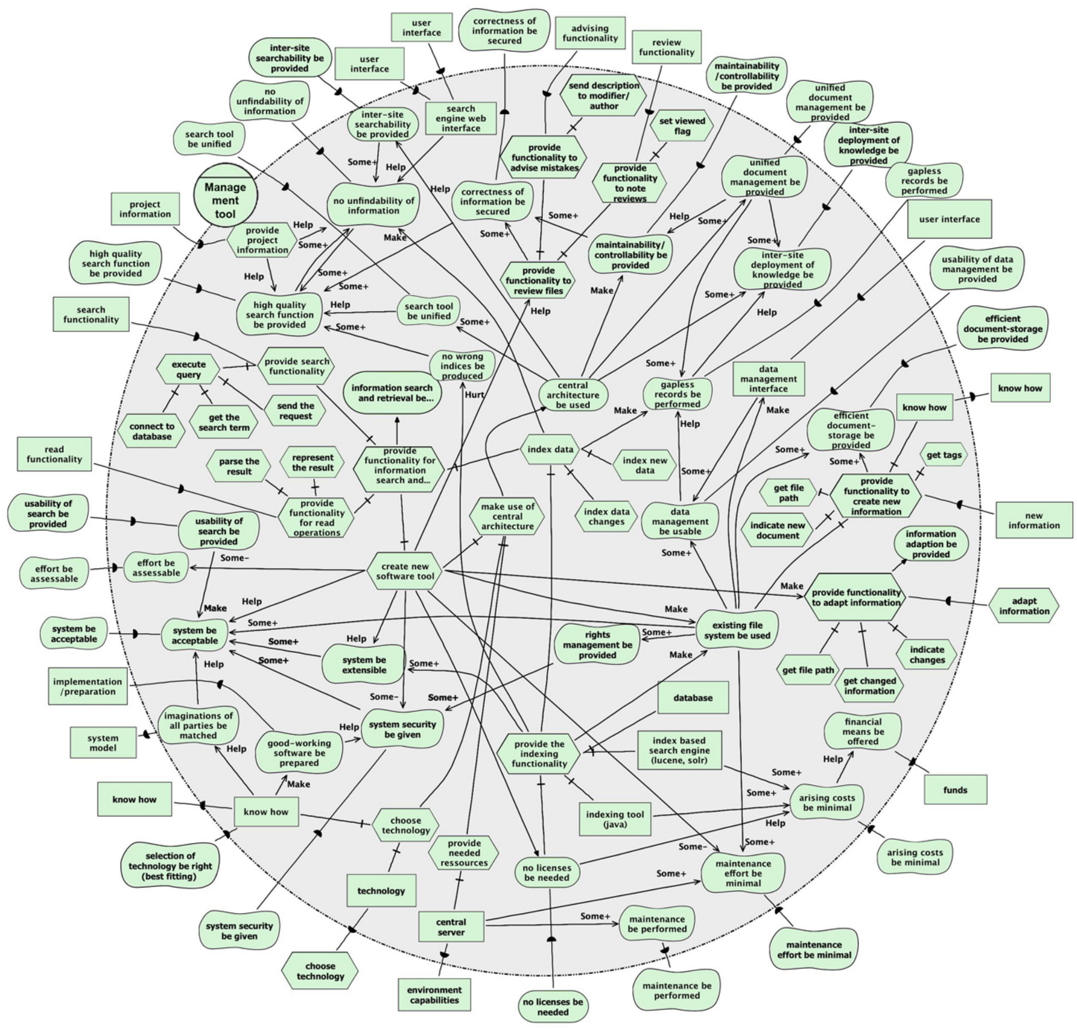

user interface
correctness of information be secured
advising functionality
review functionality
inter-site searchability be provided
user interface
maintainability /controllability be provided
unified document management be provided
no unfindability of information
search engine web interface
send description to modifier/ author
set viewed flag
inter-site searchability be provided
inter-site deployment of knowledge be provided
search tool be unified
gapless records be performed
Manage ment tool
provide functionality to advise mistakes
provide functionality to note reviews
unified document management be provided
project information
no unfindability of information
correctness of information be secured
user interface
provide project information
high quality search function be provided
maintainability/ controllability be provided
inter-site deployment of knowledge be provided
usability of data management be provided
provide functionality to review files
search functionality
high quality search function be provided
search tool be unified
efficient document-storage be provided
execute query
provide search functionality
no wrong indices be produced
information search and retrieval be...
central architecture be used
gapless records be performed
data management interface
know how
connect to database
get the search term
send the request
efficient document-storage be provided
know how
read functionality
parse the result
represent the result
provide functionality for information search and...
index data
index new data
get tags
provide functionality to create new information
usability of search be provided
provide functionality for read operations
usability of search be provided
make use of central architecture
index data changes
data management be usable
get file path
indicate new document
new information
effort be assessable
effort be assessable
create new software tool
information adaption be provided
provide functionality to adapt information
adapt information
system be acceptable
system be acceptable
rights management be provided
existing file system be used
get file path
get changed information
indicate changes
implementation /preparation
system be extensible
database
imaginations of all parties be matched
financial means be offered
system model
good-working software be prepared
system security be given
provide the indexing functionality
index based search engine (lucene, solr)
funds
know how
know how
indexing tool (java)
arising costs be minimal
choose technology
selection of technology be right (best fitting)
provide needed ressources
no licenses be needed
maintenance effort be minimal
arising costs be minimal
technology
system security be given
central server
maintenance be performed
maintenance effort be minimal
choose technology
environment capabilities
no licenses be needed
maintenance be performed
Help
Make
Hurt
Some+
Some-

Figure 3.7.: Strategic rationale sub-model for the document management system.

Each of the two approaches brings benefits on one side and drawbacks on the other side and the decision is not an easy one. The idea of coming to terms with this decision is to add each technology to the strategic rationale model (shown in figure 3.5) and to perform a goal evaluation with the algorithm described in [Strohmaier et al., 2007].

Modeling technologies require background knowledge and a deeper understanding in the functionality of each environment. By modeling both possibilities, two different SR models appear, where the internal relations of all stakeholders around the software remain the same. Also the dependencies of and to the external parties do not change. The differences between these two models lie in the way how dependencies are treated inside the management tool. The results are shown in figures 3.7 and 3.8.

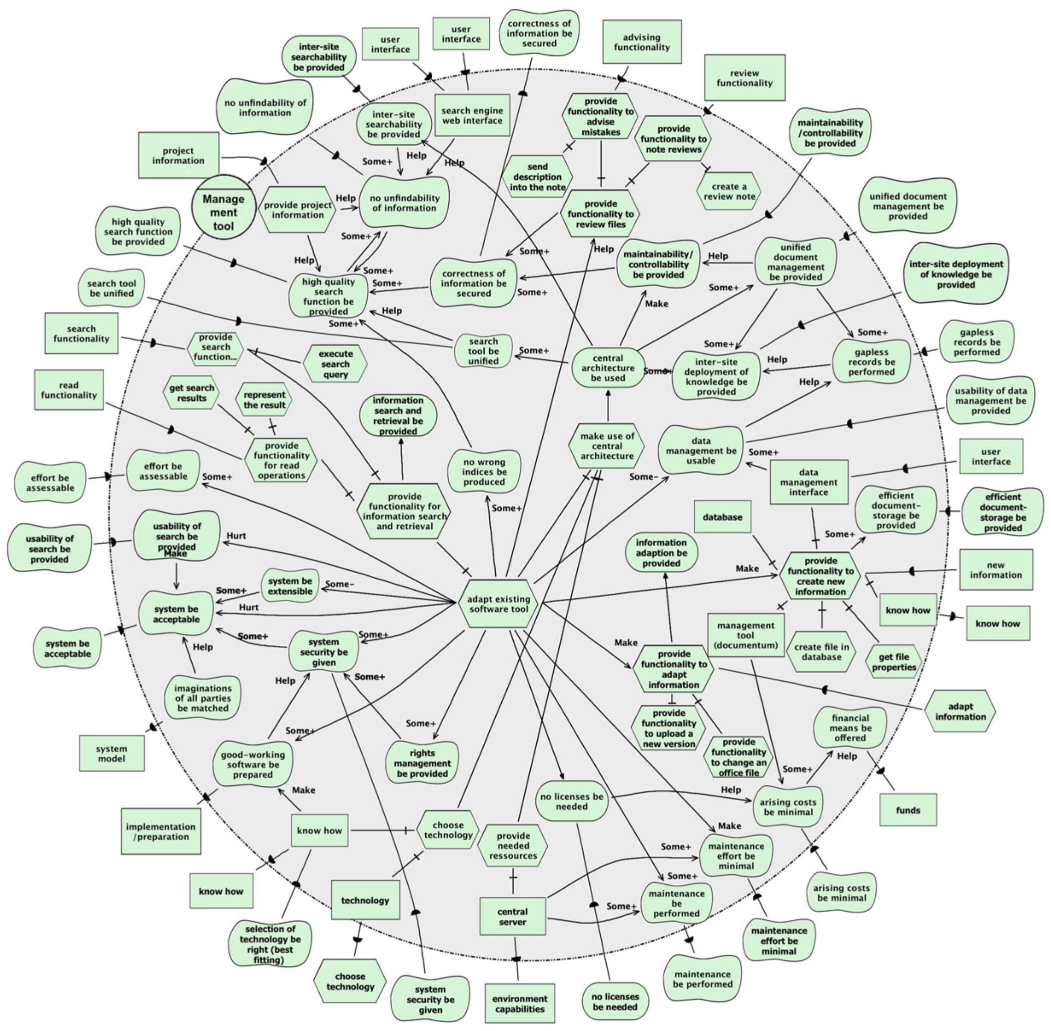


Figure 3.8.: Strategic rationale sub-model for the search engine.

3.2.2. Goal Evaluation

On implementing a huge software-based system with more than one stakeholder, a great number of requirements need to be considered. In most cases, it is absolutely impossible to fulfill exactly all the demands because some of them are contradictory. The target is to find the best fitting technology to accomplish the highest fraction of all goals and softgoal and furthermore, the most important ones. The varieties on internal relations of the software that occur between the different technologies, effect the satisfiability of goals and softgoals. To know whether the impact is positive or negative, goal evaluation - as described in chapter 2.2.3 - is performed on both models.

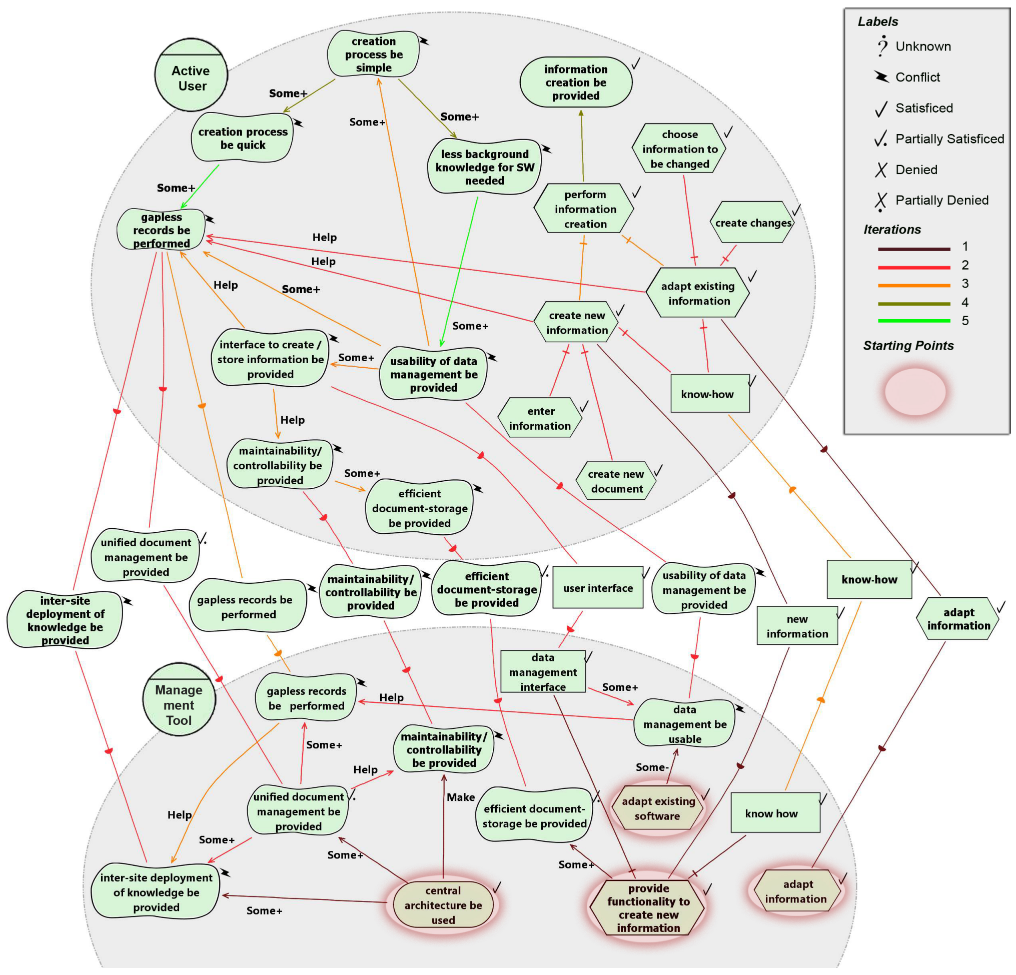

Labels
Unknown
Conflict
Satisficed
Partially Satisficed
Denied
Partially Denied
Iterations
1
2
3
4
5
Starting Points
Active User
creation process be simple
information creation be provided
creation process be quick
less background knowledge for SW needed
choose information to be changed
perform information creation
create changes
gapless records be performed
adapt existing information
create new information
interface to create / store information be provided
usability of data management be provided
know-how
enter information
create new document
maintainability/ controllability be provided
efficient document-storage be provided
unified document management be provided
inter-site deployment of knowledge be provided
gapless records be performed
maintainability/ controllability be provided
efficient document-storage be provided
user interface
usability of data management be provided
know-how
new information
adapt information
data management interface
Manage ment Tool
gapless records be performed
data management be usable
maintainability/ controllability be provided
unified document management be provided
adapt existing software
know how
efficient document-storage be provided
inter-site deployment of knowledge be provided
central architecture be used
provide functionality to create new information
adapt information
Some+
Some-
Help
Make

Figure 3.9.: Document management system sub-model - Evaluation Step One.

At the beginning, one or more goals or softgoals of the software's internal rationales are chosen. Then its influence to adjacent elements is simulated as a propagation in two steps. Modeling the individual impacts in the first propagation step is done according to the rules of figure 2.21. To make this step reasonable, it is shown in form of a sub-model in figure 3.9. This example demonstrates an extract from the evaluation of the first mentioned technology, the traditional document management system, where an already known tool is adapted to fulfill the requirements of the stakeholders. The *Active User* with its internal rationales and the *Management Tool* with a subset of internal relations are visualized.

The starting points in our sub-model are marked with red cycles. Their labels - they are all marked as *satisficed* - result from evaluating the entire internal rationales from figure 3.7. The first propagation is split up into different iterations. Each iteration handles the impact of the actual parent nodes to their child nodes. In the subsequent iteration, the child nodes become the new parents and again the influence on their child nodes are simulated. To understand how the iterations are performed, they are visualized with colored arrows and links that differ for each iteration. The legend, that can be found in the top right hand corner of the figure, shows the notation for the used labels and assigns colors to the overall five iterations.

Individual impacts can be detected easily if an element is influenced by just one other node, or all the incoming labels are of the same type. In such cases, the proper label of this element is set at the end of each run. If there are more impacts with varied meanings on one specific element, the label *conflict* is used in the first propagation.

For example the goal *central architecture be used* is a starting point in our evaluation process. In the first iteration, this node affects all of its child nodes in a positive way. After this iteration, only the softgoal *unified document management be provided* can be finally labeled and the others show conflicts. While the softgoal *gapless records be provided* further affects the first child node of the starting point, *unified document management be provided* shows additional influence on its first two children. Because of that fact, *inter-site deployment of knowledge be provided* and *maintainability be provided* are marked as conflicted. The final label for the softgoal *unified document management be provided* requires decision of the domain expert, because the some+ relation can be either a make or a help relation. In this particular case, the influence is not that strong and the *partially satisficed* label fits better.

The affects of *contribution links* can thus be distinguished by predefined rules. In the case of strategic dependencies, the label is taken over from the dependee over the dependum to the depender. To give an example, the task *provide functionality to adapt information* transfers the label *satisficed* to the resource *new information* and furthermore to the task *adapt existing information* from the *Active User.* If *decomposition links* are used, the parent is satisfied if all sub-elements are marked as *satisficed.* This happens in iteration three, where the two subtasks *create new information* and *adapt existing information* give the label *satisficed* to their parent task *perform information creation.*

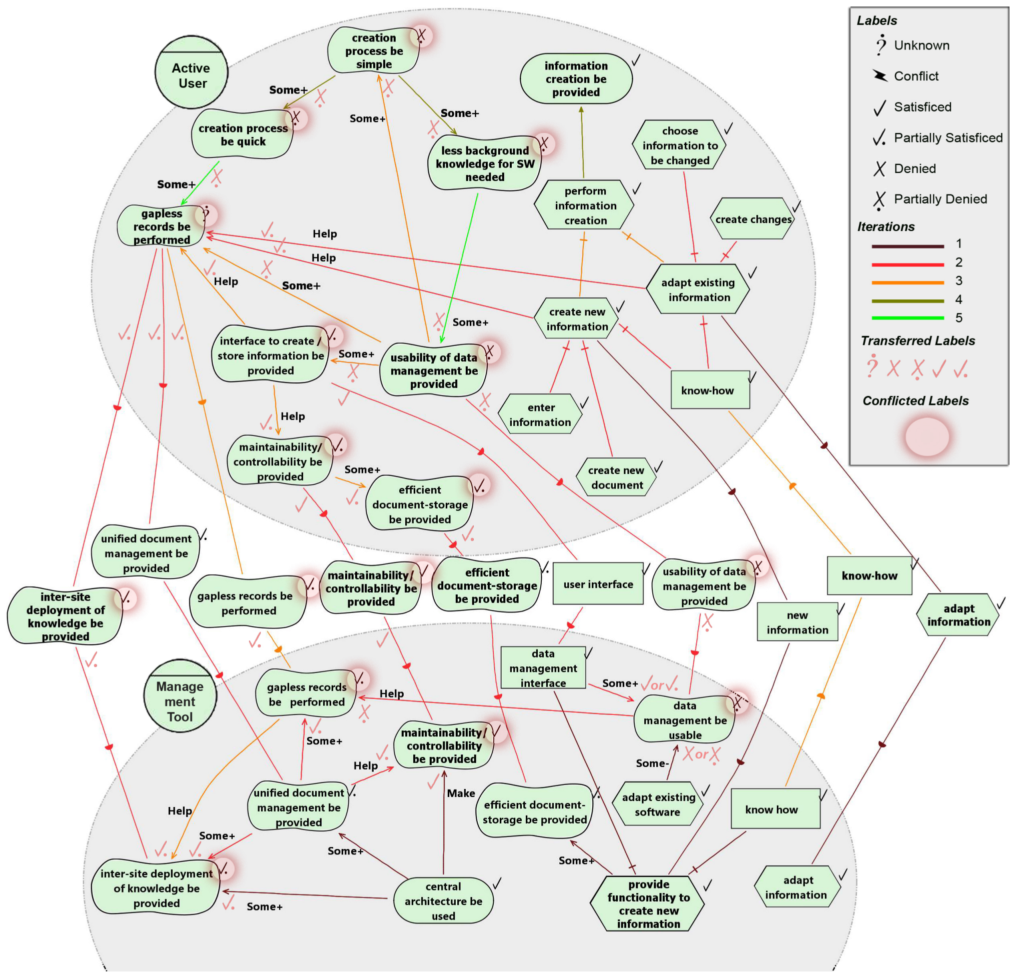

Active User
creation process be simple
information creation be provided
creation process be quick
less background knowledge for SW needed
choose information to be changed
perform information creation
create changes
gapless records be performed
adapt existing information
create new information
interface to create / store information be provided
usability of data management be provided
know-how
enter information
maintainability/ controllability be provided
efficient document-storage be provided
create new document
unified document management be provided
inter-site deployment of knowledge be provided
gapless records be performed
maintainability/ controllability be provided
efficient document-storage be provided
user interface
usability of data management be provided
know-how
new information
adapt information
data management interface
Manage ment Tool
gapless records be performed
data management be usable
maintainability/ controllability be provided
unified document management be provided
efficient document-storage be provided
adapt existing software
know how
inter-site deployment of knowledge be provided
central architecture be used
provide functionality to create new information
adapt information
Some+
Some-
Help
Make
Labels
Unknown
Conflict
Satisficed
Partially Satisficed
Denied
Partially Denied
Iterations
1
2
3
4
5
Transferred Labels
Conflicted Labels

Figure 3.10.: Document management system sub-model - Evaluation Step Two.

After modeling the individual impacts, all conflicted elements are handled. The different effects of each node are summed up in propagation step two, what can be seen in figure 3.10. The approach is again split up in five iterations, each visualized in a different color. Here, only the *conflict* labels from propagation step one are of importance - they are marked with red cycles. To get an understanding on how the labels within red cycles are set, all incoming influences to those nodes are illustrated by small red symbols at the peak of *contribution links* and upon *dependency linkages*. The domain expert needs to consider all red-marked labels before the final label can be set.

If we look at the softgoal *data management be provided*, two transferred labels can be identified. The resource *data management interface* shows a positive influence and the task *adapt existing software* a negative one upon the target softgoal. Because of the double meaning of the some+ and the some- relation, the labels *satisficed* or *partially satisficed* on one side and *denied* or *partially denied* on the other side resulted in the *conflict* in propagation step one. Providing the resource *data management interface* is indeed good for the usability of the data management. However, by adapting an existing software with a relative high complexity, instead of creating a new, simple software with the appropriate functionality, the negative impact on the softgoal *usability of data management be provided* is higher than the positive one. Therefore the softgoal *data management be usable* is marked as *partially denied* and this label is transferred via the dependum *usability of data management be provided* to the internal rationale of the dependee.

Such assessment is performed for every single conflicted label to see the final results of the goal evaluation. The internal softgoal *gapless records be performed* is a special case in propagation step two. All together, there are eight influences to this node, six are positive, modeled with the *partially satisficed* label and two are negative, indicated with the *partially denied* label. One could say that the resulting label is a compelling case because the relation from positive to negative is like six to two, but under consideration of the domain knowledge, the negative effects from a missing usability and a time-consuming creation process lead to an *unknown* labeled softgoal.

3.2.3. Technology Decision

The decision between two technologies is normally made according to the analysis of the entire results of the goal evaluation. For the choice in our sub-scenario, the goal evaluation for the second mentioned technology, the search engine running on an intranet site together with a helper tool to provide information, is still missing. The approach on this evaluation is on the analogy of those from the document management system, visualized in figures 3.9 and 3.10. Therefore, only the results after both propagation steps are shown for the second technology in figure 3.11.

To highlight the different impacts on the satisfaction of stakeholders' demands that are caused by the different technologies, the internal elements are marked with colored cycles. Goals, softgoals, resources and tasks with the same label as outcome in both technologies are marked with a blue cycle, signifying a neutral position in contrast to the compared sub-model. Improvements are identified with green and degradations relating to figure 3.10 with red cycles.

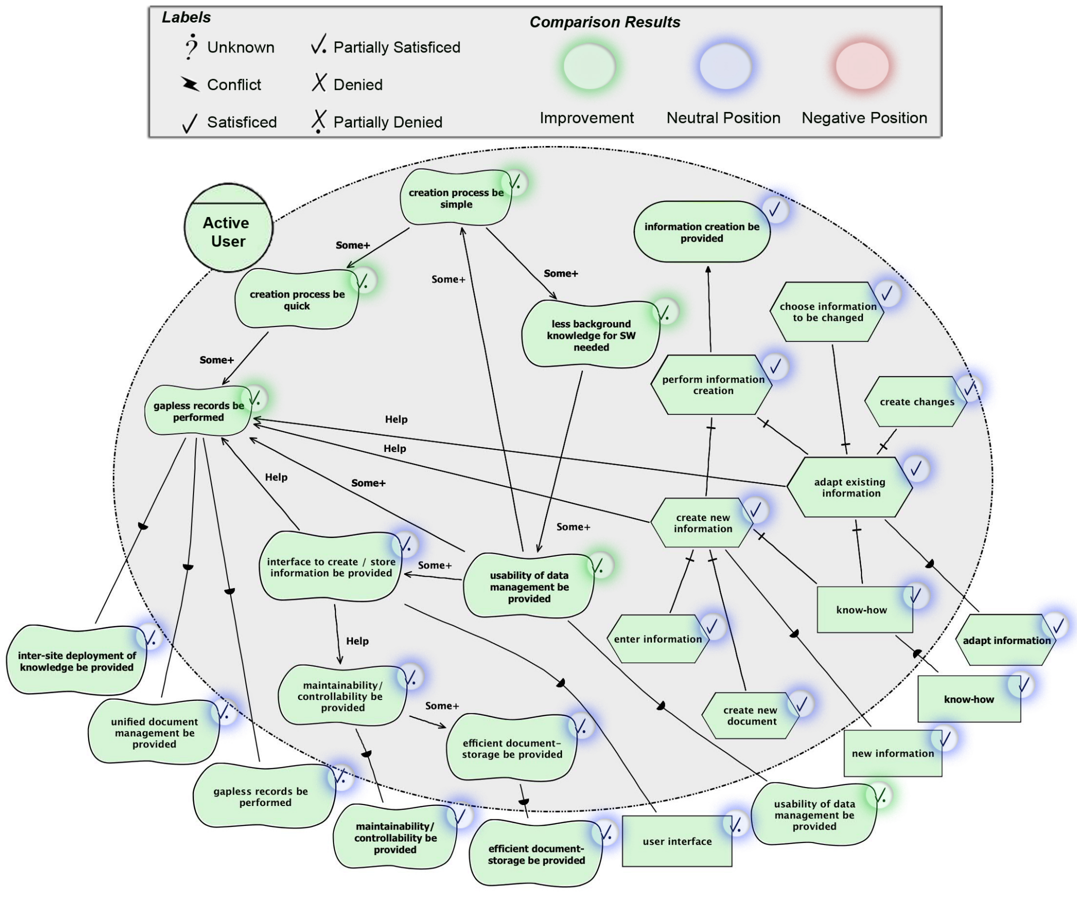

Labels
Unknown
Partially Satisficed
Conflict
Denied
Satisficed
Partially Denied
Comparison Results
Improvement
Neutral Position
Negative Position
Active User
creation process be simple
information creation be provided
creation process be quick
less background knowledge for SW needed
choose information to be changed
perform information creation
create changes
gapless records be performed
adapt existing information
create new information
interface to create / store information be provided
usability of data management be provided
know-how
enter information
adapt information
inter-site deployment of knowledge be provided
maintainability/ controllability be provided
create new document
unified document management be provided
efficient document-storage be provided
new information
gapless records be performed
maintainability/ controllability be provided
efficient document-storage be provided
user interface
usability of data management be provided
Some+
Help

Figure 3.11.: Crawler sub-model - Evaluation Results.

In our example scenario the technology choice can be greatly facilitated by the goal evaluation. The second technology shows no degradations in the polarity of all internal elements. A big part of them has the same positive label meaning that these requirements can be fulfilled by both technologies. There are also some softgoals that can only be accomplished if the second technology is chosen:

- *usability of data management be provided*
- *less background knowledge for SW needed*
- *creation process be simple*
- *creation process be quick*

The achievement of the softgoal *gapless records be performed* is *unknown* for technology one, but *satisficed* for technology two. The insight of this evaluation advises to realize the *Management Tool* by implementing a new software-based system consisting of a search engine with a proper search index and a helper tool that fills the index.

The goal evaluation of the complete SR models also shows clear benefits on choosing this technology. As a result, technology two is used to realize this project. Further information on the goal evaluation can be found in the appendix. Chapter A lists the results of other stakeholders in form of several sub-models. Comparisons between the possible approaches are again visualized with colored cycles.

4. Implementation

This chapter shows how the software is implemented and the whole system is realized whilst taking into account requirements of the different stakeholders and guidelines as well as restrictions appearing within the company. It is described how the chosen technology is used and how the different parts of the software are structured and furthermore used. To understand the interaction of different components and the communication between them, figures are used.

4.1. Architecture

The architecture of the system can be described by figure 4.1. The main components are the following:

- Crawler: the Crawler is an independent Java application that provides information for the search index. It reads configuration information out of proper files and iterates through all listed project paths to calculate and send important metadata of every contained file to the server.
- GUI: the GUI that is supplied in form of an intranet page contains two main functionalities. Search requests with various parameters need to be performed and additional file manipulation is offered to handle milestone specific functionality, like calculating the project progress or exerting influence on the file status by assigning or reviewing documents.
- Tomcat Server: both components, the Crawler and the GUI, communicate via HTTP to the Tomcat server, that offers its functionality in form of servlets. All together, four of them exit. The entire functionality is divided into different categories according to its purpose. Such servlet handles different types of input and chooses the right treatment. Thereby, it communicates to the search index.
- Search Index: the search index is realized with the enterprise search platform Apache Solr. This open source project is driven by the "Apache Software Foundation" and it offers easy access to the functionality of an extended

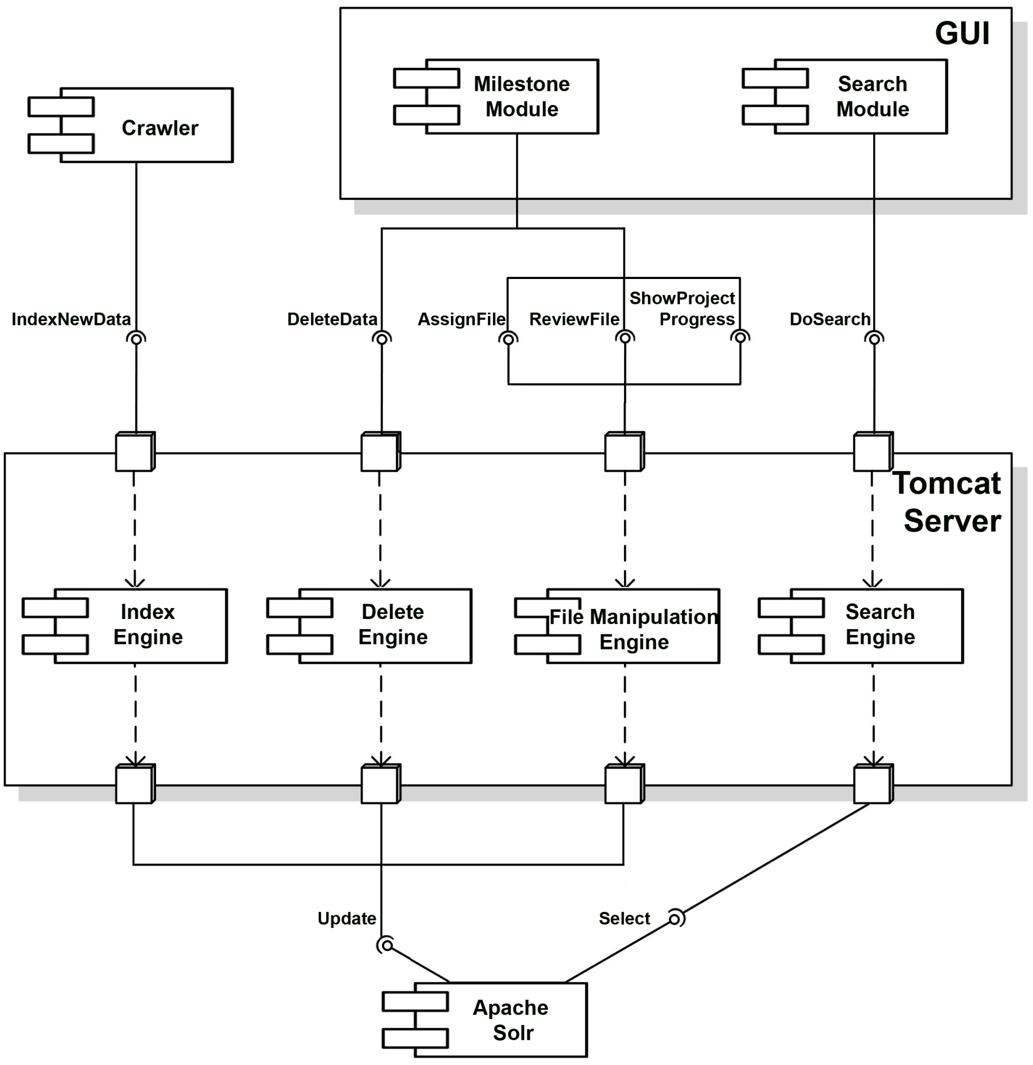


Figure 4.1.: Component diagram.

full text search. The storage of needed information is also performed with the help of the Solr application.

4.2. File Crawler

To provide information to the search index, the crawler regularly runs on the existing network drives to capture information. The list of projects that need to be observed is outsourced to a configuration file and read in on application start-up. Every time a new project is introduced by test engineers, a new line is added to this document. Its structure is described by table 4.1.

[Project Name];	[Absolute Path]
M1234;	\\drive1\M1234\
M2312;	\\drive1\M2312\
M4233;	\\drive3\M4233\
...	

Table 4.1.: Structure of the configuration file of the crawler.

The crawler touches all files inside the predefined projects, calculates needed information and transfers it within a HTTP request to the *IndexEngine*. Additional, milestone-specific information is stored in separate property documents when a file is assigned to a milestone or it is released by a software admin. If such document exists for a given filename, also the additional information is read out and added to the HTTP request.

This Java application makes use of some data classes that define routines to calculate information out of files and properties. A helper class is also included, which holds the entire definitions and useful information that is used by more than one other class. All of these additional classes are used by the server application as well. This avoids double held code and improves the maintainability of the whole system. The structure of the crawler application is described by the class diagram of figure 4.2. The data and helper classes get described in detail in section 4.4. Its structure is shown by figures 4.5 and 4.6.

To ensure the regular execution of the program and furthermore the up-to-dateness of the search index, the crawling tool is exported as a runnable ".jar file". The internal job server of the company starts this file daily.

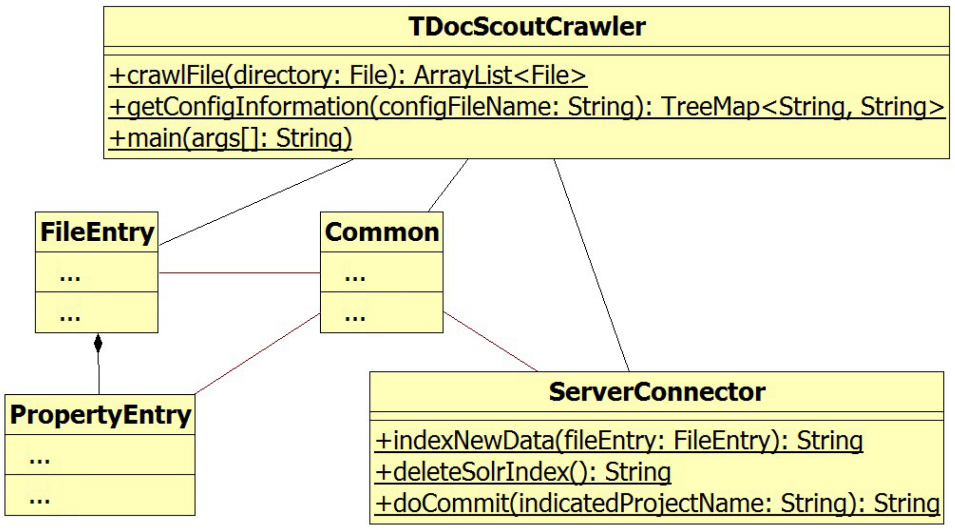

TDocScoutCrawler
+crawlFile(directory: File): ArrayList<File>
+getConfigInformation(configFileName: String): TreeMap<String, String>
+main(args[]: String)
FileEntry
Common
PropertyEntry
ServerConnector
+indexNewData(fileEntry: FileEntry): String
+deleteSolrIndex(): String
+doCommit(indicatedProjectName: String): String

Figure 4.2.: Class diagram of the file crawler application.

- *TDocScoutCrawler*: this class is the core element of the crawler application. It contains functionality to read out the crawler configuration and to walk through all files of the given file paths. The communication with the server is also initiated here.
- *ServerConnector*: the *ServerConnector* holds functionality to communicate with the servlets of the Tomcat server. It is used to indicate new file information, delete the complete index and commit changes to the index. Project names are also registered through this class and the possibility to indicate a reload of the server configuration is given.

4.3. Graphical User Interface (GUI)

The GUI is realized with Java Server Pages (JSP) to create dynamic HTML code. It is accessible via the intranet platform from any location of the Infineon Technologies AG. Layout and functionality of this page are optimized for Internet Explorer 9 because this is the common browser in the company. The design is adjusted to the company standards to fit to other intranet sites.

The heart of the user interface is the index.jsp that holds the functionality for search queries as well as those for milestone specific actions. The interface consists of different HTML div-elements, that are styled with Cascading Style Sheets (CSS). If an action is triggered by the user, the interface communicates via HTTP requests to the proper servlet. After realizing the requested event, the response is visualized.

The search functionality is used by entering a search term and pressing the "start search" button. If a project is selected, the scope of the retrieval process is restricted from the entire projects to this specific one. The search results contain information about the name, path and type of each file. The search term can be used to create own queries by using terms from the Apache Lucene parser syntax (described in [Carlson, 2006]). This gives the user freedom to create an individual search request, what improves the findability of information.

If the project progress is prompted, needed information is requested from the *FileProcessingEngine*. It is defined for each milestone what files are needed to complete it. The interface visualizes the file status with colors.

4.4. Tomcat Server

The Apache Tomcat server is already introduced in the company and administrated by the IT department. Any access underlies certain restrictions and can only be realized via tickets from the so-called "HelpDesk" platform. This server provides the environment for the enterprise application, that in turn represents the logical part of the management tool. The entire functionality is grouped to four servlets - derived from the Java class "javax.servlet.http.HttpServlet" - according to its purpose. A servlet handles different types of input and chooses the right treatment. Thereby it communicates to the search index via the HTTP protocol. In most cases it additionally produces a proper response and redirects it to the user interface. The structure of the server application is visualized in figure 4.3 in form of a class diagram. Descriptions of the blank classes are shown in figures 4.4, 4.5 and 4.6.

- *IndexEngine*: the *IndexEngine* reads out the type of action that needs to be performed. Thereby it distinguishes between indexing new information in form of a FileEntry object, updating the list of indicated project names and reloading the server configuration.
- *FileProcessingEngine*: this servlet is needed to calculate and furthermore show the project progress by providing milestone information. It detects files that are assigned to a project and figures out its file status that can be *not assigned*, *unreleased* or *released*. The *FileProcessingEngine* is also able to execute new assignments of test documents to milestones and file reviews. Permissions to assigned files are regulated through the internal

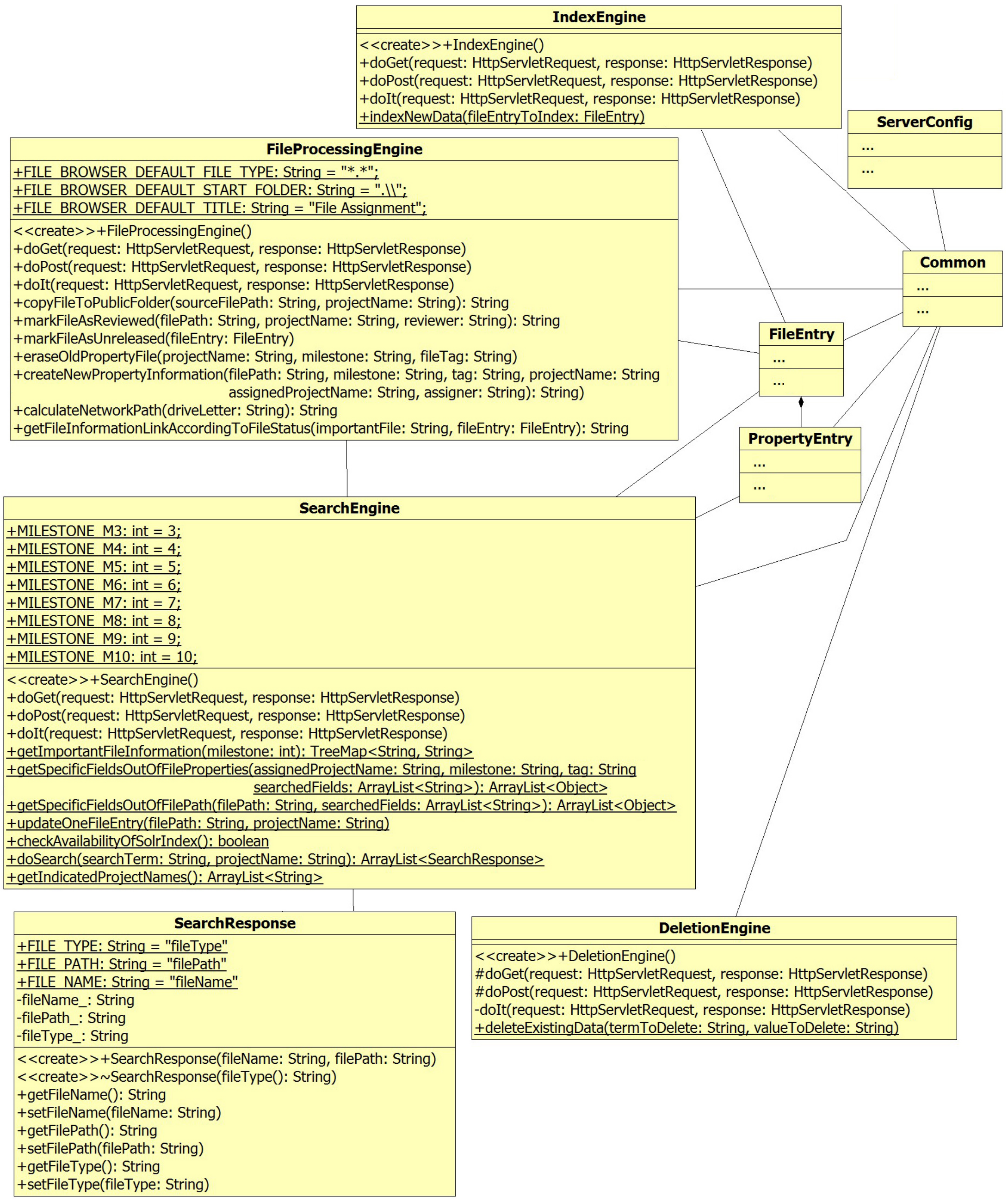
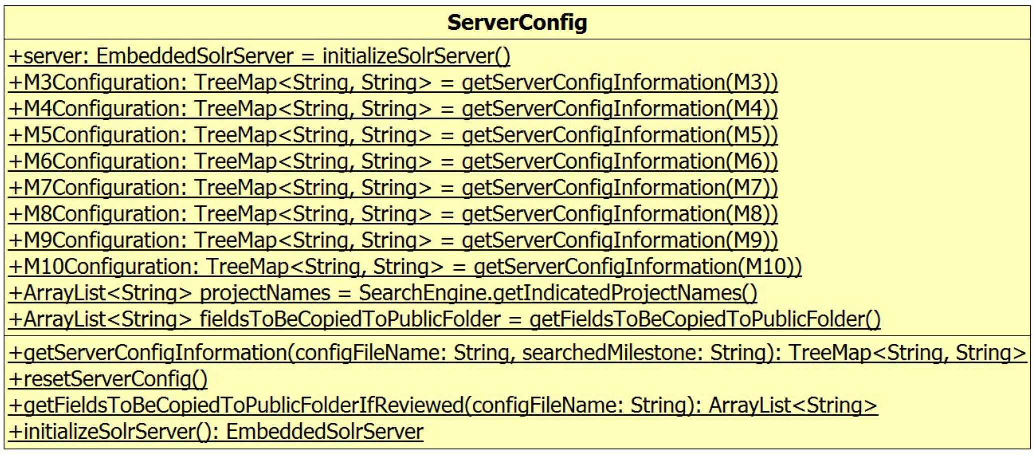

IndexEngine
<<create>>+IndexEngine()
+doGet(request: HttpServletRequest, response: HttpServletResponse)
+doPost(request: HttpServletRequest, response: HttpServletResponse)
+doIt(request: HttpServletRequest, response: HttpServletResponse)
+indexNewData(fileEntryToIndex: FileEntry)
ServerConfig
...
...
FileProcessingEngine
+FILE_BROWSER_DEFAULT_FILE_TYPE: String = "*.*";
+FILE_BROWSER_DEFAULT_START_FOLDER: String = ".\\";
+FILE_BROWSER_DEFAULT_TITLE: String = "File Assignment";
<<create>>+FileProcessingEngine()
+doGet(request: HttpServletRequest, response: HttpServletResponse)
+doPost(request: HttpServletRequest, response: HttpServletResponse)
+doIt(request: HttpServletRequest, response: HttpServletResponse)
+copyFileToPublicFolder(sourceFilePath: String, projectName: String): String
+markFileAsReviewed(filePath: String, projectName: String, reviewer: String): String
+markFileAsUnreleased(fileEntry: FileEntry)
+eraseOldPropertyFile(projectName: String, milestone: String, fileTag: String)
+createNewPropertyInformation(filePath: String, milestone: String, tag: String, projectName: String
assignedProjectName: String, assigner: String): String)
+calculateNetworkPath(driveLetter: String): String
+getFileInformationLinkAccordingToFileStatus(importantFile: String, fileEntry: FileEntry): String
Common
...
...
FileEntry
...
...
PropertyEntry
...
...
SearchEngine
+MILESTONE_M3: int = 3;
+MILESTONE_M4: int = 4;
+MILESTONE_M5: int = 5;
+MILESTONE_M6: int = 6;
+MILESTONE_M7: int = 7;
+MILESTONE_M8: int = 8;
+MILESTONE_M9: int = 9;
+MILESTONE_M10: int = 10;
<<create>>+SearchEngine()
+doGet(request: HttpServletRequest, response: HttpServletResponse)
+doPost(request: HttpServletRequest, response: HttpServletResponse)
+doIt(request: HttpServletRequest, response: HttpServletResponse)
+getImportantFileInformation(milestone: int): TreeMap<String, String>
+getSpecificFieldsOutOfFileProperties(assignedProjectName: String, milestone: String, tag: String
searchedFields: ArrayList<String>): ArrayList<Object>
+getSpecificFieldsOutOfFilePath(filePath: String, searchedFields: ArrayList<String>): ArrayList<Object>
+updateOneFileEntry(filePath: String, projectName: String)
+checkAvailabilityOfSolrIndex(): boolean
+doSearch(searchTerm: String, projectName: String): ArrayList<SearchResponse>
+getIndicatedProjectNames(): ArrayList<String>
SearchResponse
+FILE_TYPE: String = "fileType"
+FILE_PATH: String = "filePath"
+FILE_NAME: String = "fileName"
-fileName_: String
-filePath_: String
-fileType_: String
<<create>>+SearchResponse(fileName: String, filePath: String)
<<create>>~SearchResponse(fileType(): String)
+getFileName(): String
+setFileName(fileName: String)
+getFilePath(): String
+setFilePath(filePath: String)
+getFileType(): String
+setFileType(fileType: String)
DeletionEngine
<<create>>+DeletionEngine()
#doGet(request: HttpServletRequest, response: HttpServletResponse)
#doPost(request: HttpServletRequest, response: HttpServletResponse)
-doIt(request: HttpServletRequest, response: HttpServletResponse)
+deleteExistingData(termToDelete: String, valueToDelete: String)

Figure 4.3.: Class diagram server application.

right management of the company and therefore are the same as those on network drives. Sometimes read access for released files is desired to be provided for every user of the intranet site. Therefore a public folder is introduced that ensures unrestricted read access to all comprised files. Moreover this servlet is responsible for storing a copy of selected files to this folder when they get released.

- *SearchEngine*: the *SearchEngine* is responsible for conducting regular and restricted, project-specific search requests. Therefore, it communicates with the search index to retrieve needed information. The response is redirected to the user interface in form of instances of the class SearchResponse. The servlet also contains functionality to check the availability of the search index and to read out the list of indicated project names as well as required files of each milestone and to inquire specific fields from a search index entry.
- *DeletionEngine*: this servlet is called to delete specific fields or the whole search index. It reads out the requested type of action and processes it without redirecting a response to the GUI.
- *SearchResponse*: the *SearchResponse* is a helper class, that is used by the *SearchEngine*. It bundles information that is important for the result of a regular search request, in particular name, path and type of one file.



```
ServerConfig
+server: EmbeddedSolrServer = initializeSolrServer()
+M3Configuration: TreeMap<String, String> = getServerConfigInformation(M3))
+M4Configuration: TreeMap<String, String> = getServerConfigInformation(M4))
+M5Configuration: TreeMap<String, String> = getServerConfigInformation(M5))
+M6Configuration: TreeMap<String, String> = getServerConfigInformation(M6))
+M7Configuration: TreeMap<String, String> = getServerConfigInformation(M7))
+M8Configuration: TreeMap<String, String> = getServerConfigInformation(M8))
+M9Configuration: TreeMap<String, String> = getServerConfigInformation(M9))
+M10Configuration: TreeMap<String, String> = getServerConfigInformation(M10))
+ArrayList<String> projectNames = SearchEngine.getIndicatedProjectNames()
+ArrayList<String> fieldsToBeCopiedToPublicFolder = getFieldsToBeCopiedToPublicFolder()
+getServerConfigInformation(configFileName: String, searchedMilestone: String): TreeMap<String, String>
+resetServerConfig()
+getFieldsToBeCopiedToPublicFolderIfReviewed(configFileName: String): ArrayList<String>
+initializeSolrServer(): EmbeddedSolrServer
```

Figure 4.4.: Class details ServerConfig.

- *ServerConfig*: the configuration information for the server part is outsourced to a proper file. It contains information about important or required files for each milestone and gives the possibility to provide read access for every user, independent of their rights on the network file system. The *ServerConfig* class reads out this configuration information and holds it as members. It also requests the list of indicated project names and provides functionality to reset the configuration. Finally, this class holds and initializes an object of the class EmbeddedSolrServer, that represents the Apache Solr index in its embedded version. Further information on the search index is given in section 4.5.

Table 4.2 describes the structure of the server configuration file. First, the milestone is stated, followed by the description of the important file, that is also shown in the GUI. The unique "Tag" field is needed for internal use. The last parameter is optional and only added to public releases.

[Milestone];	[Description];	[Tag];	[True(optional)]
M3;	Test Concept M3;	file-M3-1;	true
M3;	TCAF;	file-M3-2;	
M4;	Test Concept M4;	file-M4-1;	
M4;	Tester Resource Overview BE;	file-M4-2;	
M4;	Tester Resource Overview FE;	file-M4-3;	
M4;	Test Time Roadmap / Estimation;	file-M4-4;	true
...			

Table 4.2.: Structure of the configuration file of the server.

Beside the crawler, the server application also makes use of the classes *FileEntry*, *PropertyEntry* and *Common* from the "data" package.

- *FileEntry*: this data class represents the entry of one single file from the search index. The crawler creates one *FileEntry* object for each file in his search scope. All contained members are calculated and filled out automatically by the constructor. This object is then transported to the *IndexEngine* servlet, that induces the storage of the information. A *PropertyEntry* object is also added as a member that contains information that is read from the property file.
- *PropertyEntry*: a *PropertyEntry* represents the additional data that needs to be stored once the status gets changed by an assignment or release of an already indicated file. If such property information is available, it is read out automatically and added to the appropriate FileEntry. Java properties are used to store the additional information on the file system. They are created by the *FileProcessingEngine*.

- *Common*: *Common* represents a helper class with many constant definitions; among others, paths to configuration and log files, parameter names for the HTTP transfer, formats for storing and visualizing dates, error descriptions and return messages are defined. It also provides the functionality to create output to the logfile and to construct error messages in HTML format.

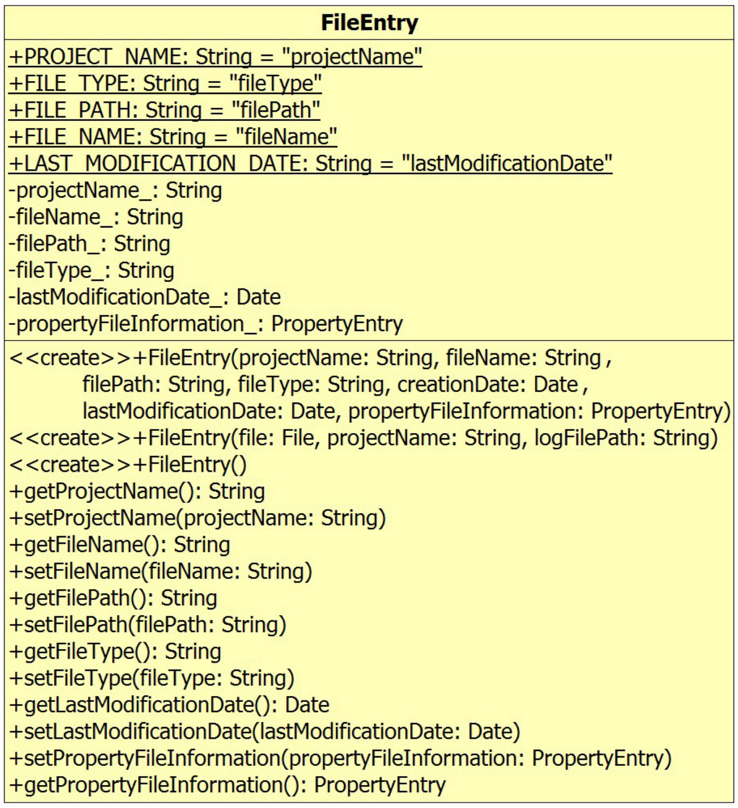
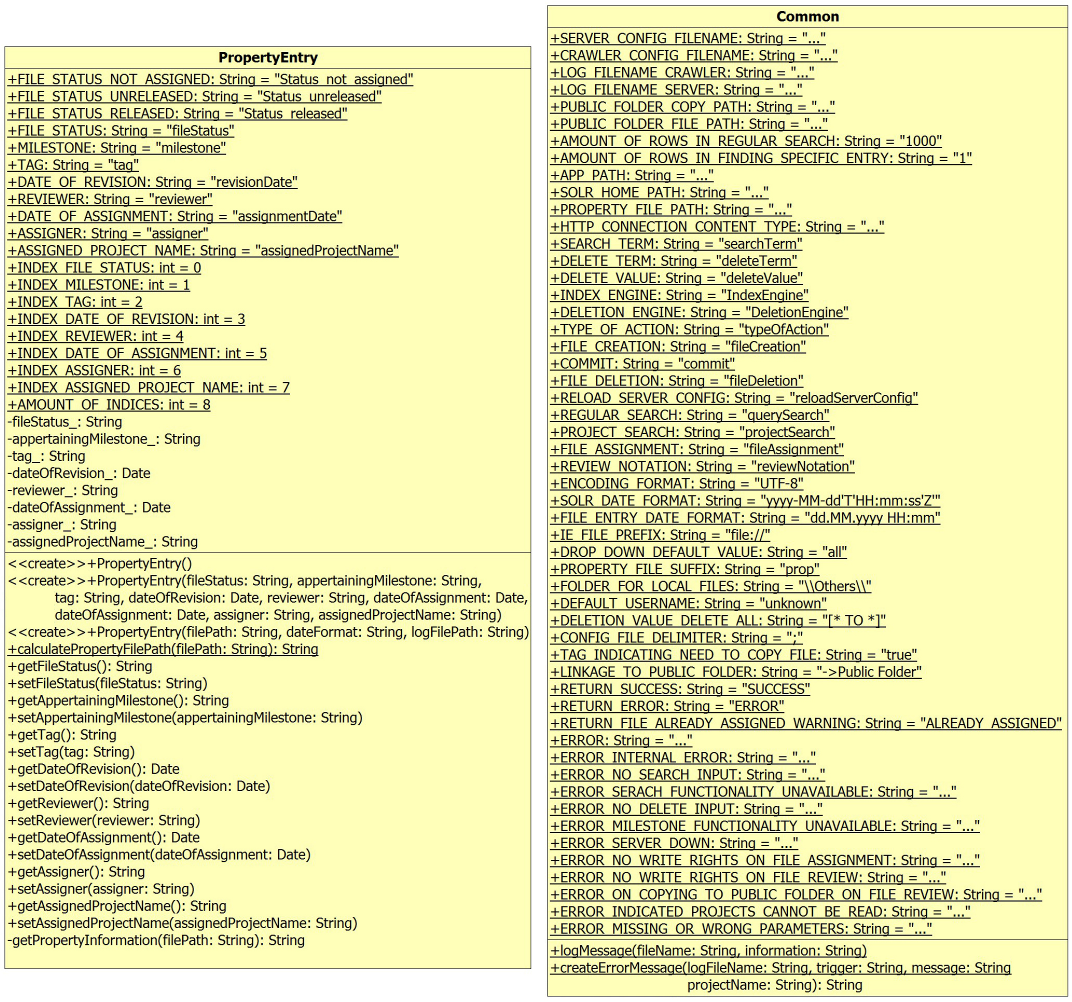

FileEntry
+PROJECT_NAME: String = "projectName"
+FILE_TYPE: String = "fileType"
+FILE_PATH: String = "filePath"
+FILE_NAME: String = "fileName"
+LAST_MODIFICATION_DATE: String = "lastModificationDate"
-projectName_: String
-fileName_: String
-filePath_: String
-fileType_: String
-lastModificationDate_: Date
-propertyFileInformation_: PropertyEntry
<<create>>+FileEntry(projectName: String, fileName: String, filePath: String, fileType: String, creationDate: Date, lastModificationDate: Date, propertyFileInformation: PropertyEntry)
<<create>>+FileEntry(file: File, projectName: String, logFilePath: String)
<<create>>+FileEntry()
+getProjectName(): String
+setProjectName(projectName: String)
+getFileName(): String
+setFileName(fileName: String)
+getFilePath(): String
+setFilePath(filePath: String)
+getFileType(): String
+setFileType(fileType: String)
+getLastModificationDate(): Date
+setLastModificationDate(lastModificationDate: Date)
+setPropertyFileInformation(propertyFileInformation: PropertyEntry)
+getPropertyFileInformation(): PropertyEntry

Figure 4.5.: Class details FileEntry.

4.5. Search Index

Apache Solr is based on the previously introduced and perhaps better known "Lucene Java Library". It offers a powerful full-text search functionality with lots of additional advantages. Different interfaces are provided to communicate with other applications. In this project, the HTTP API is used to implement information exchange between the servlets and the search index via the Hypertext Transfer Protocol. Solr needs a Java Servlet Container like the Apache Tomcat, to be runnable. In general it is used as a separate application. In this project, the concept of "Embedded Solr" is implemented, where no separate Java process is needed and the functionality can be used inside the management tool through the Solrj Java API [Efendi et al., 2007].



```
PropertyEntry
+FILE_STATUS_NOT_ASSIGNED: String = "Status_not_assigned"
+FILE_STATUS_UNRELEASED: String = "Status_unreleased"
+FILE_STATUS_RELEASED: String = "Status_released"
+FILE_STATUS: String = "fileStatus"
+MILESTONE: String = "milestone"
+TAG: String = "tag"
+DATE_OF_REVISION: String = "revisionDate"
+REVIEWER: String = "reviewer"
+DATE_OF_ASSIGNMENT: String = "assignmentDate"
+ASSIGNER: String = "assigner"
+ASSIGNED_PROJECT_NAME: String = "assignedProjectName"
+INDEX_FILE_STATUS: int = 0
+INDEX_MILESTONE: int = 1
+INDEX_TAG: int = 2
+INDEX_DATE_OF_REVISION: int = 3
+INDEX_REVIEWER: int = 4
+INDEX_DATE_OF_ASSIGNMENT: int = 5
+INDEX_ASSIGNER: int = 6
+INDEX_ASSIGNED_PROJECT_NAME: int = 7
+AMOUNT_OF_INDICES: int = 8
-fileStatus_: String
-appertainingMilestone_: String
-tag_: String
-dateOfRevision_: Date
-reviewer_: String
-dateOfAssignment_: Date
-assigner_: String
-assignedProjectName_: String
<<create>>+PropertyEntry()
<<create>>+PropertyEntry(fileStatus: String, appertainingMilestone: String,
        tag: String, dateOfRevision: Date, reviewer: String, dateOfAssignment: Date,
        dateOfAssignment: Date, assigner: String, assignedProjectName: String)
<<create>>+PropertyEntry(filePath: String, dateFormat: String, logFilePath: String)
+calculatePropertyFilePath(filePath: String): String
+getFileStatus(): String
+setFileStatus(fileStatus: String)
+getAppertainingMilestone(): String
+setAppertainingMilestone(appertainingMilestone: String)
+getTag(): String
+setTag(tag: String)
+getDateOfRevision(): Date
+setDateOfRevision(dateOfRevision: Date)
+getReviewer(): String
+setReviewer(reviewer: String)
+getDateOfAssignment(): Date
+setDateOfAssignment(dateOfAssignment: Date)
+getAssigner(): String
+setAssigner(assigner: String)
+getAssignedProjectName(): String
+setAssignedProjectName(assignedProjectName: String)
-getPropertyInformation(filePath: String): String
```

```
Common
+SERVER_CONFIG_FILENAME: String = "..."
+CRAWLER_CONFIG_FILENAME: String = "..."
+LOG_FILENAME_CRAWLER: String = "..."
+LOG_FILENAME_SERVER: String = "..."
+PUBLIC_FOLDER_COPY_PATH: String = "..."
+PUBLIC_FOLDER_FILE_PATH: String = "..."
+AMOUNT_OF_ROWS_IN_REGULAR_SEARCH: String = "1000"
+AMOUNT_OF_ROWS_IN_FINDING_SPECIFIC_ENTRY: String = "1"
+APP_PATH: String = "..."
+SOLR_HOME_PATH: String = "..."
+PROPERTY_FILE_PATH: String = "..."
+HTTP_CONNECTION_CONTENT_TYPE: String = "..."
+SEARCH_TERM: String = "searchTerm"
+DELETE_TERM: String = "deleteTerm"
+DELETE_VALUE: String = "deleteValue"
+INDEX_ENGINE: String = "IndexEngine"
+DELETION_ENGINE: String = "DeletionEngine"
+TYPE_OF_ACTION: String = "typeOfAction"
+FILE_CREATION: String = "fileCreation"
+COMMIT: String = "commit"
+FILE_DELETION: String = "fileDeletion"
+RELOAD_SERVER_CONFIG: String = "reloadServerConfig"
+REGULAR_SEARCH: String = "querySearch"
+PROJECT_SEARCH: String = "projectSearch"
+FILE_ASSIGNMENT: String = "fileAssignment"
+REVIEW_NOTATION: String = "reviewNotation"
+ENCODING_FORMAT: String = "UTF-8"
+SOLR_DATE_FORMAT: String = "yyyy-MM-dd'T'HH:mm:ss'Z'"
+FILE_ENTRY_DATE_FORMAT: String = "dd.MM.yyyy HH:mm"
+IE_FILE_PREFIX: String = "file://"
+DROP_DOWN_DEFAULT_VALUE: String = "all"
+PROPERTY_FILE_SUFFIX: String = "prop"
+FOLDER_FOR_LOCAL_FILES: String = "\\Others\\"
+DEFAULT_USERNAME: String = "unknown"
+DELETION_VALUE_DELETE_ALL: String = "[* TO *]"
+CONFIG_FILE_DELIMITER: String = ";"
+TAG_INDICATING_NEED_TO_COPY_FILE: String = "true"
+LINKAGE_TO_PUBLIC_FOLDER: String = "->Public Folder"
+RETURN_SUCCESS: String = "SUCCESS"
+RETURN_ERROR: String = "ERROR"
+RETURN_FILE_ALREADY_ASSIGNED_WARNING: String = "ALREADY_ASSIGNED"
+ERROR: String = "..."
+ERROR_INTERNAL_ERROR: String = "..."
+ERROR_NO_SEARCH_INPUT: String = "..."
+ERROR_SERACH_FUNCTIONALITY_UNAVAILABLE: String = "..."
+ERROR_NO_DELETE_INPUT: String = "..."
+ERROR_MILESTONE_FUNCTIONALITY_UNAVAILABLE: String = "..."
+ERROR_SERVER_DOWN: String = "..."
+ERROR_NO_WRITE_RIGHTS_ON_FILE_ASSIGNMENT: String = "..."
+ERROR_NO_WRITE_RIGHTS_ON_FILE_REVIEW: String = "..."
+ERROR_ON_COPYING_TO_PUBLIC_FOLDER_ON_FILE_REVIEW: String = "..."
+ERROR_INDICATED_PROJECTS_CANNOT_BE_READ: String = "..."
+ERROR_MISSING_OR_WRONG_PARAMETERS: String = "..."
+logMessage(fileName: String, information: String)
+createErrorMessage(logFileName: String, trigger: String, message: String
                projectName: String): String
```

Figure 4.6.: Class details PropertyEntry and Common.

The data storage is also provided by this innovation. Indicated information is archived in different file segments and stored on the web space of the server application. The data is stored and represented in XML format. Further information about the search platform Solr and descriptions and tutorial on how to install and use it, can be found in [Foundation, 2007a] and [Foundation, 2007b], as well as in [Hatcher et al., 2011]. The usage of Solrj is shown in [Ryan et al., 2011].

5. Results and Validation

In this chapter the results of the project are presented. The final management tool is described in detail and visible parts are presented with graphics. Procedures are justified by comparing parts of the software with the selected requirements.

5.1. Requirement Categories

The functionality of the resulting software is based on the set of requirements from different stakeholders. To make the analysis and argumentation less complex, the requirements are grouped in categories. Their realization is then shown and further described. Dependencies from figure 3.3 and internal rationales from figure 3.5 are used to create the list of categories. The following elements are contained:

- Search functionality: a unique search tool is needed, that offers the possibility for an inter-site information retrieval. A simple and usable interface should be provided, that delivers and represents high quality results. Information about certain projects should be distinguishable to others and an overview of all available projects should be given.
- Project progress: the progress of each test project should be visualized for users. This information is needed by employees or sites that work on different milestones within the same project or are somehow involved to it. People in authority want to be up to date relating to problems and delays appearing inside the project progress to stick to the planned deadlines.
- Data storage: the information storage should be as simple as possible. Employees must not be confused by introducing new processes just to store and use documents of any type. It is highly appreciated that all files can be found on the familiar network drives and are accessible via the windows explorer.

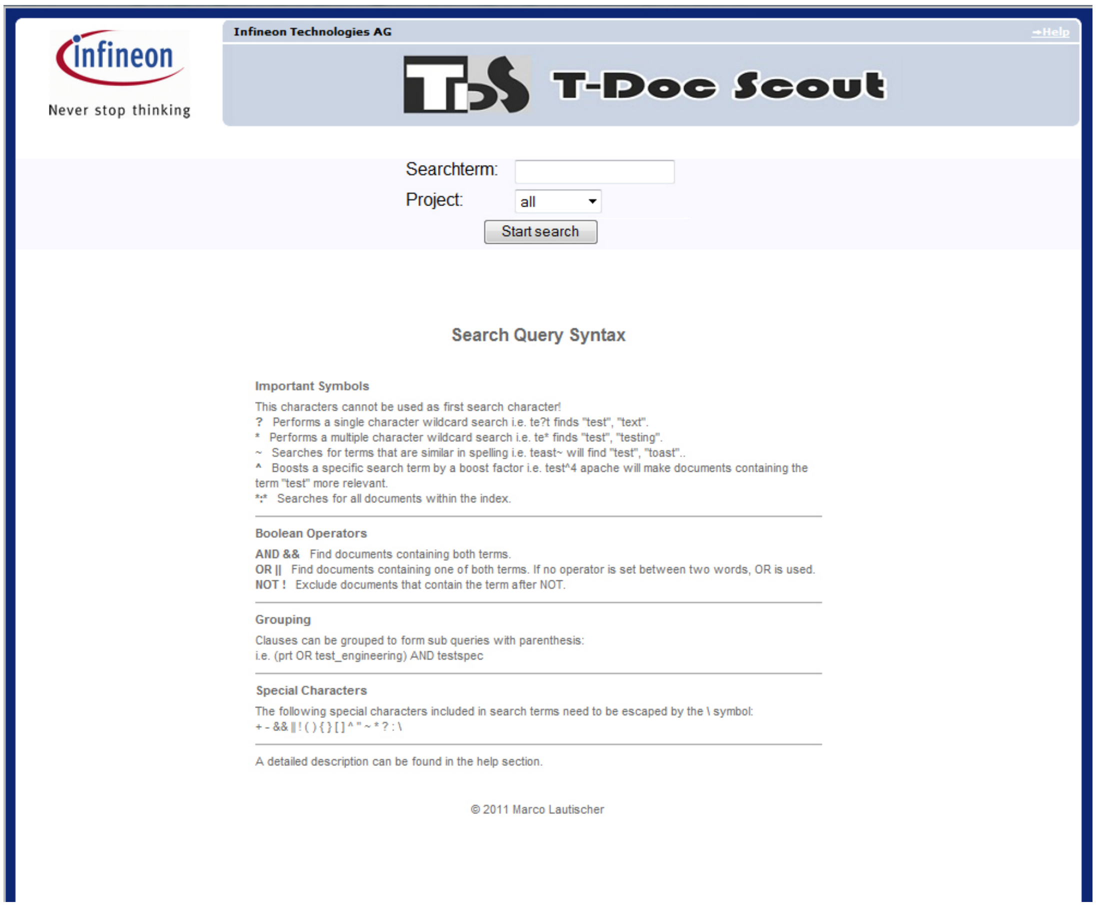
- Information creation: creating and modifying information within the software should be an easy procedure where a simple user interface is required. It must be possible to deploy knowledge from different sites. Nothing should stand in the way to perform continuous and gapless records to keep the information up to date.
- Review functionality: especially superiors and project managers desire a possibility to review added information and to ensure correctness of the content. Reviewed or released files must be distinguishable from others and clearly visualized.
- Rights management: different access rights on the content of the management system should be provided for user groups. Sensitive project data must not become accessible for restricted users due to leaks in the rights management of the software. The assignment of permissions is desired to take less time and all procedures must be realizable easily and quickly.

5.1.1. Search Functionality

The *search functionality* is realized by a graphical user interface to provide the resource *user interface[search]*. To fulfill the goal *possibility for inter-site search be provided* the GUI is realized in form of an intranet page of the company. The internal network is accessible for different sites and ensures a central entrance to the content of the management tool (softgoal *search tool be unified*). As already mentioned in the last chapters, the style of this site is adjusted to the standards of the company and is constructed in a very simple way and no unimportant functionality is included. This accomplishes the softgoal *usability of search be provided*. The information content is restricted so that a brief summary of the syntax that can be used inside the search term to increase the quality of the results (softgoal *search functionality be of high quality*). To ensure a *quick retrieval process*, the file index is kept rather small and only relevant data and file links are stored. Figure 5.1 shows the layout of the GUI.

A link to the internal Wiki is added on the top right hand side of the page. By following this link, users are able to find information on how to use the new document management system; also the search syntax is described more precisely and examples are shown there. Instructions to create the required files of each milestone, and furthermore to complete it, are also part of this Wiki, that helps to satisfy the softgoal *no background knowledge needed*.

The user has the possibility to retrieve information of the entire indicated data or to restrict the scope to a desired project. Therefore, he changes the value of



Search Query Syntax

Important Symbols

This characters cannot be used as first search character!
? Performs a single character wildcard search i.e. te?t finds "test", "text".
* Performs a multiple character wildcard search i.e. te* finds "test", "testing".
~ Searches for terms that are similar in spelling i.e. teast~ will find "test", "toast"..
^ Boosts a specific search term by a boost factor i.e. test^4 apache will make documents containing the term "test" more relevant.
: Searches for all documents within the index.

Boolean Operators

AND && Find documents containing both terms.
OR || Find documents containing one of both terms. If no operator is set between two words, OR is used.
NOT ! Exclude documents that contain the term after NOT.

Grouping

Clauses can be grouped to form sub queries with parenthesis:
i.e. (prt OR test_engineering) AND testspec

Special Characters

The following special characters included in search terms need to be escaped by the \ symbol:
+ - && || ! ( ) { } [ ] ^ " ~ * ? : \

A detailed description can be found in the help section.

© 2011 Marco Lautischer

Figure 5.1.: Screenshot User Interface - Home.

the dropdown list that indicates the actual project, what is shown in figure 5.2. The default value "all" adds no additional information to the search request and the query is evaluated on the whole content of the search index. If a project is selected, only project specific information can be encountered (resource *project specific information*).

In the final implementation, the amount of calculated search results is limited to 1000 to avoid long waiting times if a very common search term causes a huge amount of hits. It is also redundant to produce a higher amount of results because nobody wants to go through all of them. Moreover, in such cases, restrictions are added by using elements of the defined search syntax. This maximum can be easily changed to any other value.

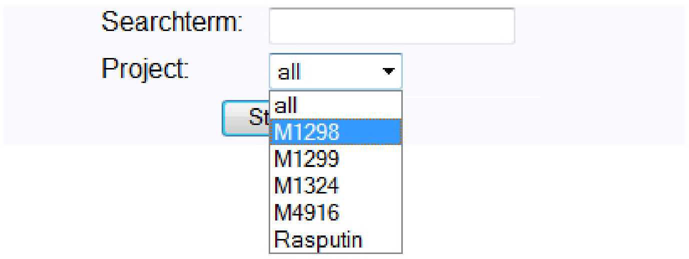
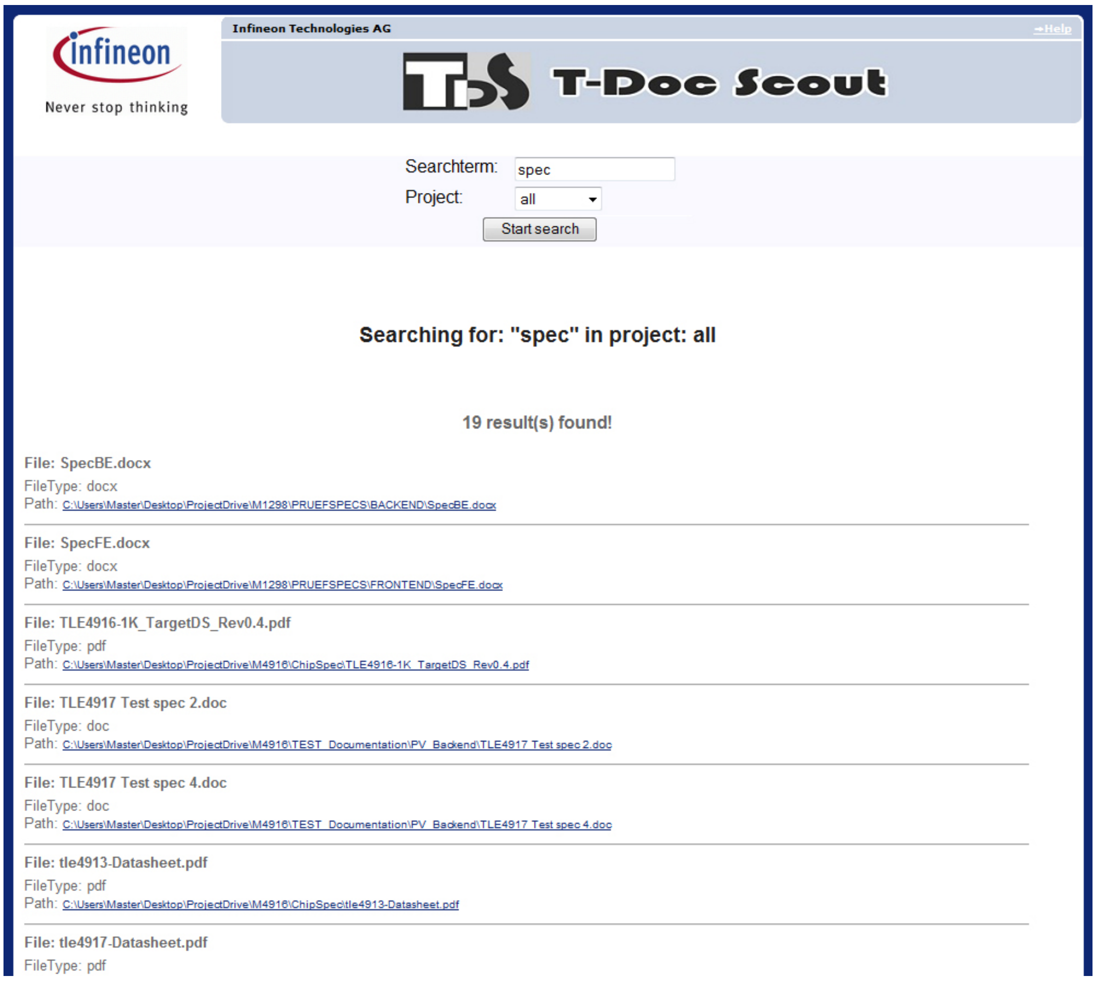


Figure 5.2.: Screenshot GUI - Project choice.

The resource *information representation* is provided in form of search results that consist of three parts. The first element is the filename; it is highlighted because in many cases this information is enough to identify the desired dataset. After that, the file type is listed; in common cases it just contains the proper ending. For unknown file types a mapping can be performed so that a short description gives the user information about the structure of the result. To give an example, a file that ends on ".tsf" contains specific test results. The file type in that case could be: "Test Results". Finally the absolute file path is shown in form of a link. This information helps users to choose the needed data and gives him a quick access to it. Figure 5.3 and 5.4 visualize the search functionality by searching for the term *spec* in the entire index as well as in project "M1298".



Searchterm: spec
Project: all

Searching for: "spec" in project: all

19 result(s) found!

File: SpecBE.docx
FileType: docx
Path: C:\Users\Master\Desktop\ProjectDrive\M1298\PRUEFSPECS\BACKEND\SpecBE.docx

File: SpecFE.docx
FileType: docx
Path: C:\Users\Master\Desktop\ProjectDrive\M1298\PRUEFSPECS\FRONTEND\SpecFE.docx

File: TLE4916-1K_TargetDS_Rev0.4.pdf
FileType: pdf
Path: C:\Users\Master\Desktop\ProjectDrive\M4916\ChipSpec\TLE4916-1K_TargetDS_Rev0.4.pdf

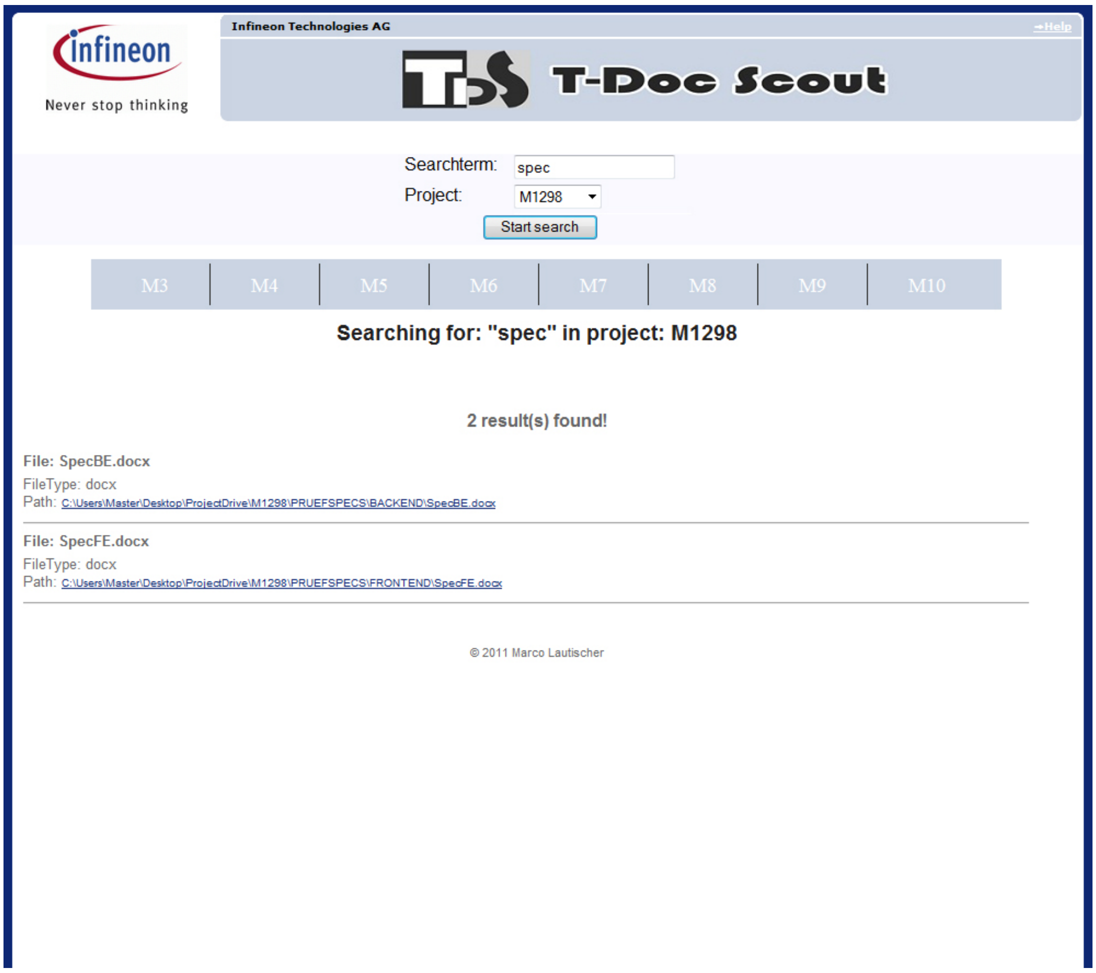
File: TLE4917 Test spec 2.doc
FileType: doc
Path: C:\Users\Master\Desktop\ProjectDrive\M4916\TEST_Documentation\PV_Backend\TLE4917 Test spec 2.doc

File: TLE4917 Test spec 4.doc
FileType: doc
Path: C:\Users\Master\Desktop\ProjectDrive\M4916\TEST_Documentation\PV_Backend\TLE4917 Test spec 4.doc

File: tle4913-Datasheet.pdf
FileType: pdf
Path: C:\Users\Master\Desktop\ProjectDrive\M4916\ChipSpec\tle4913-Datasheet.pdf

File: tle4917-Datasheet.pdf
FileType: pdf

Figure 5.3.: Screenshot GUI - Regular search.



Searching for: "spec" in project: M1298

2 result(s) found!

File: SpecBE.docx
FileType: docx
Path: C:\Users\Master\Desktop\ProjectDrive\M1298\PRUEFSPECS\BACKEND\SpecBE.docx

File: SpecFE.docx
FileType: docx
Path: C:\Users\Master\Desktop\ProjectDrive\M1298\PRUEFSPECS\FRONTEND\SpecFE.docx

© 2011 Marco Lautischer

Figure 5.4.: Screenshot GUI - Project specific search.

5.1.2. Project Progress

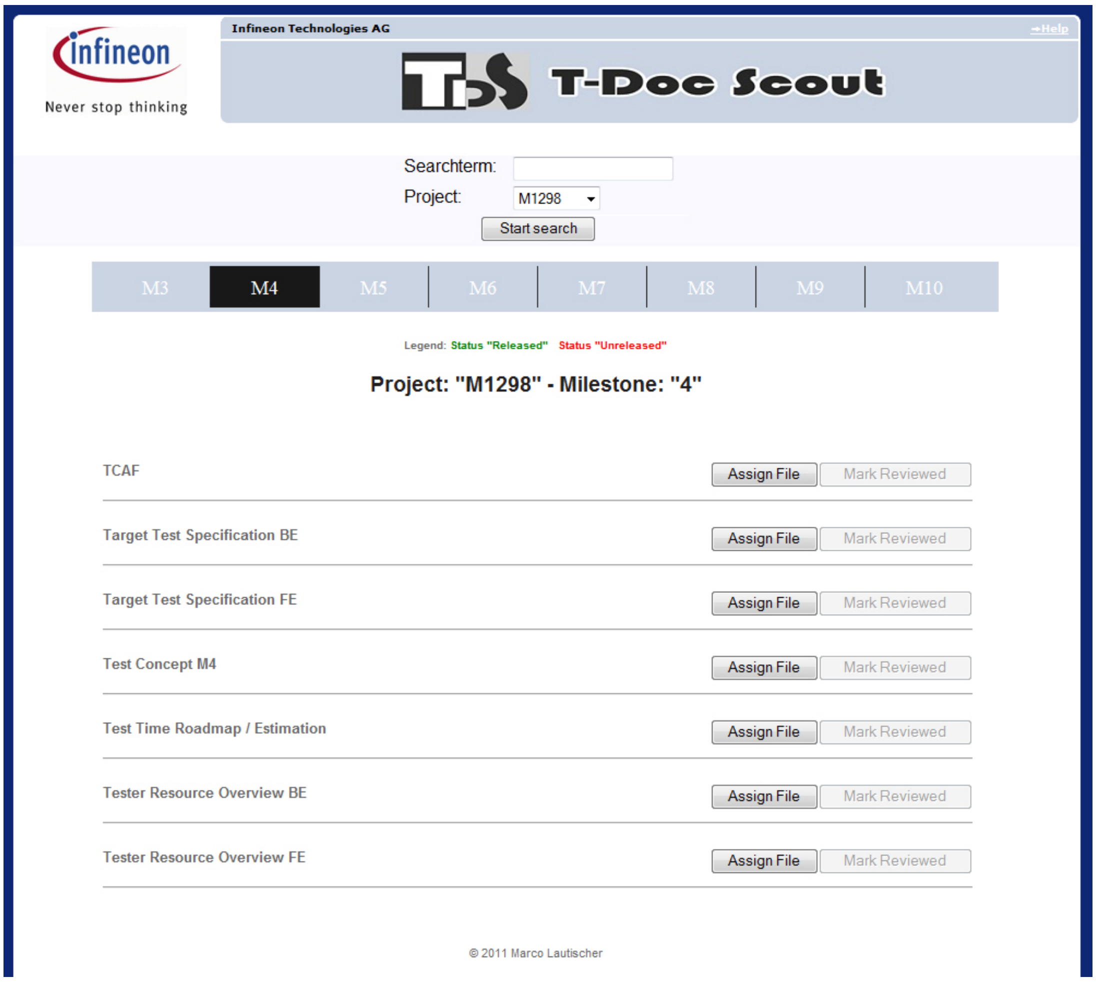
Every time a project is selected via the dropdown menu, the proper milestone bar appears, that satisfies the softgoal *project progress be shown*. The first two milestones are different and no required files exist for them, so they are removed and number three to nine are visualized. Each entry of the menu bar shows the list of important files. This information is stored in the server configuration and is loaded on server startup. Figure 5.5 presents the progress of the project "M1298", where milestone four is at an initial state.

The resource *monitor functionality for the project progress* is provided by adding colored file statuses. They give an overview about the project progress and help to identify problems. If no data is assigned, the status "not assigned" is indicated by a dark gray color as regular text. The proper button enables the relation of concrete information to an important file. If data is assigned, a button to perform reviews appears.

5.1.3. Data Storage

The way of storing test data in the company is not altered by using this application. Test engineers are able to use the same common network drives to store their project information to satisfy the softgoal *usability of data management be provided*. Every document can be directly accessed, modified or deleted via the file system (resource *user interface [add/modify]*) to provide the basis for gapless records (softgoal *gapless records be performed*). The new software-based system supplies additional features and helps to visualize and manage the projects. Changes on the file system are updated to the search index after the next crawler run. In the first runs inside the company, the interval is set to "daily", but it can easily be changed when needed. If files get assigned to milestone, the search index is adapted immediately. After the assignment of new files that are not in the scope of the crawler, they can nonetheless be found via the GUI.

The use of network drives as data storage is efficient and accomplishes the softgoal *efficient document-storage be provided*; no further effort in maintenance is needed (softgoal *maintenance effort be minimal*). The IT department is responsible for the enlargement of memory, if it becomes tight. Only the storage of project folders that are entered to the configuration file can be correlated with the new management tool. The task *adapt information* is accomplished by the possibilities to adapt existing information on network drives and to assign files.


Infineon Technologies AG

→Help

T-Doc Scout

Searchterm:

Project: M1298

Start search

Legend: Status "Released" Status "Unreleased"

Project: "M1298" - Milestone: "4"

TCAF	Assign File	Mark Reviewed
Target Test Specification BE	Assign File	Mark Reviewed
Target Test Specification FE	Assign File	Mark Reviewed
Test Concept M4	Assign File	Mark Reviewed
Test Time Roadmap / Estimation	Assign File	Mark Reviewed
Tester Resource Overview BE	Assign File	Mark Reviewed
Tester Resource Overview FE	Assign File	Mark Reviewed

© 2011 Marco Lautischer

Figure 5.5.: Screenshot GUI - Project progress.

5.1.4. Information Creation

The resource *new information* can be provided in two ways, by indicating file specific information through the crawler application and by file assignments caused by active users. The scope of the crawler is defined in the configuration file. If new projects are started, their names and absolute paths need to be added here. The content of the new project is indicated on the next iteration of the crawler. File assignments are performed by users of the system via the intranet page of the user interface, what accomplishes the softgoal *unified document management be provided*. The possibility for inter-site knowledge deployment is also given and the softgoal *inter-site deployment of knowledge be provided* is fulfilled.

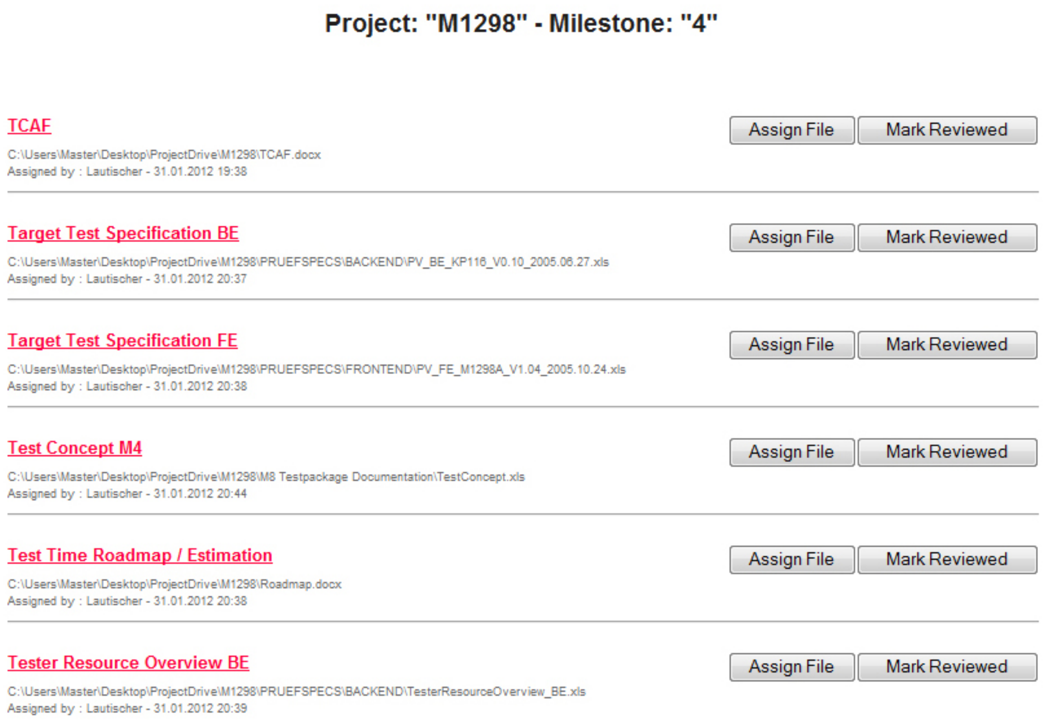
The button "Assign File" is used to relate the proper information to a required file. A browser window appears and users can choose their preferred file. The server then automatically adds property information in form of the actual timestamp and the username of the person that triggers this action. The status of the file changes to "unreleased" and is indicated with a red colored link by the GUI. After data is assigned, the button to perform the review is enabled. Property information is stored in a property file to announce it to the crawler application on the next startup and indicated to the search index as well.

The assignment of files is presented in figure 5.6. It shows the same milestone as figure 5.5 but this time in an advanced stage where required information is created and assigned properly.

5.1.5. Review Functionality

Before a milestone can be finalized, all required files must be released to fulfill the softgoal *correctness of information be secured*. The resource *review functionality* is provided in form of the "review" button inside the GUI (resource *user interface [revision]*) and the proper file status. Only justified users are allowed to review a milestone specific file, whose status becomes "released" - indicated by a green link. If an already released file gets changed again, the status is set back to "unreleased" and the *Revisor* agent must perform another content check. Advises due to wrong content are given to authors per email or in person (softgoal *possibility to make advises be given*).

It is not allowed that people review files that they have produced on their own. To realize this restriction, the username of the assigner must be different to those of the reviewer. Additional property information is added, namely the timestamp



Project: "M1298" - Milestone: "4"

TCAF
C:\Users\Master\Desktop\ProjectDrive\M1298\TCAF.docx
Assigned by : Lautischer - 31.01.2012 19:38

Target Test Specification BE
C:\Users\Master\Desktop\ProjectDrive\M1298\PRUEFSPECS\BACKEND\PV_BE_KP116_V0.10_2005.06.27.xls
Assigned by : Lautischer - 31.01.2012 20:37

Target Test Specification FE
C:\Users\Master\Desktop\ProjectDrive\M1298\PRUEFSPECS\FRONTEND\PV_FE_M1298A_V1.04_2005.10.24.xls
Assigned by : Lautischer - 31.01.2012 20:38

Test Concept M4
C:\Users\Master\Desktop\ProjectDrive\M1298\M8 Testpackage Documentation\TestConcept.xls
Assigned by : Lautischer - 31.01.2012 20:44

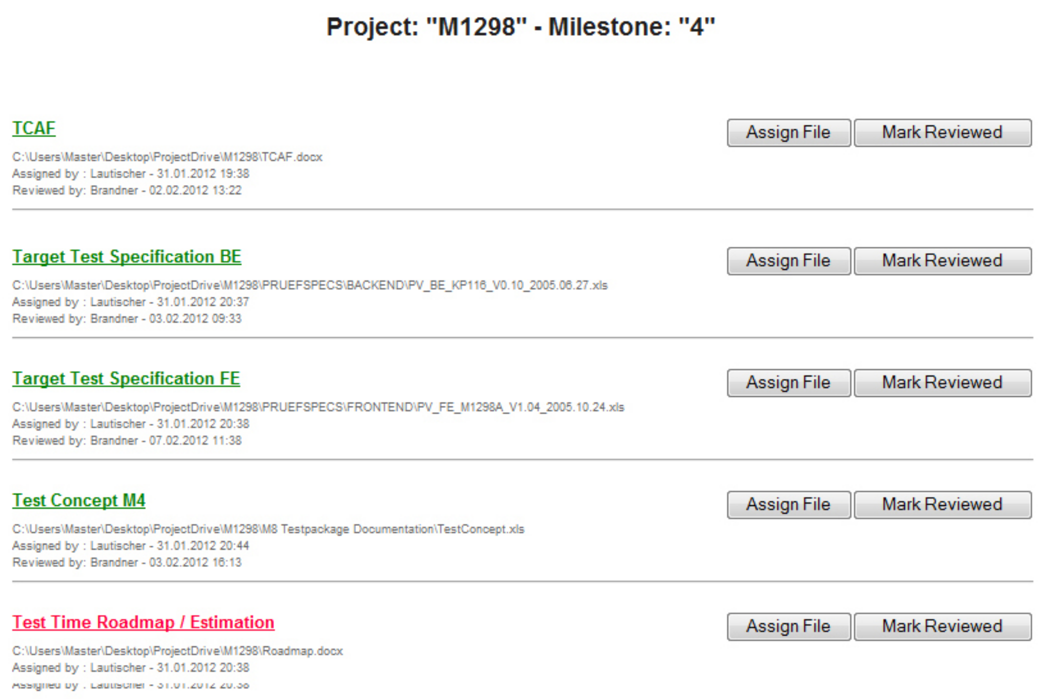
Test Time Roadmap / Estimation
C:\Users\Master\Desktop\ProjectDrive\M1298\Roadmap.docx
Assigned by : Lautischer - 31.01.2012 20:38

Tester Resource Overview BE
C:\Users\Master\Desktop\ProjectDrive\M1298\PRUEFSPECS\BACKEND\TesterResourceOverview_BE.xls
Assigned by : Lautischer - 31.01.2012 20:39

Figure 5.6.: Screenshot GUI - Assignment of files.

of the review and the name of the reviewer. The file status is also adapted to the property file and inside the search index.

In figure 5.7, most of the required information of milestone four is released. The review of the "Test Time Roadmap / Estimation" is still missing. After this review, the milestone is completed and the project progress moves on to milestone five.



Project: "M1298" - Milestone: "4"

TCAF
C:\Users\Master\Desktop\ProjectDrive\M1298\TCAF.docx
Assigned by : Lautischer - 31.01.2012 19:38
Reviewed by: Brandner - 02.02.2012 13:22

Target Test Specification BE
C:\Users\Master\Desktop\ProjectDrive\M1298\PRUEFSPECS\BACKEND\PV_BE_KP116_V0.10_2005.06.27.xls
Assigned by : Lautischer - 31.01.2012 20:37
Reviewed by: Brandner - 03.02.2012 09:33

Target Test Specification FE
C:\Users\Master\Desktop\ProjectDrive\M1298\PRUEFSPECS\FRONTEND\PV_FE_M1298A_V1.04_2005.10.24.xls
Assigned by : Lautischer - 31.01.2012 20:38
Reviewed by: Brandner - 07.02.2012 11:38

Test Concept M4
C:\Users\Master\Desktop\ProjectDrive\M1298\M8 Testpackage Documentation\TestConcept.xls
Assigned by : Lautischer - 31.01.2012 20:44
Reviewed by: Brandner - 03.02.2012 16:13

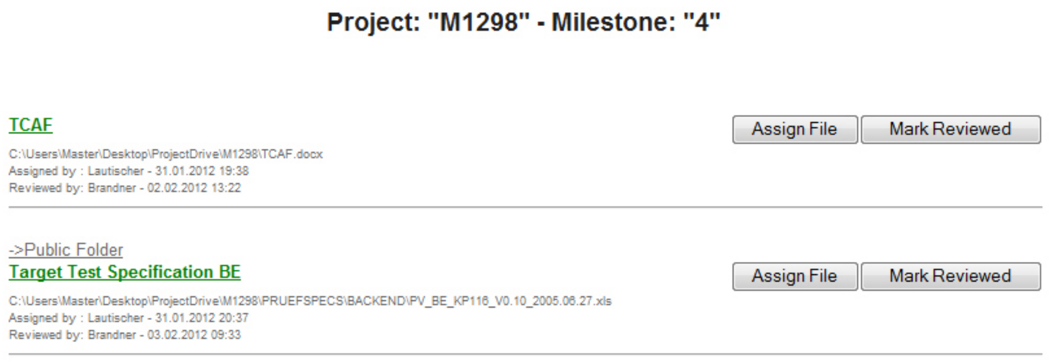
Test Time Roadmap / Estimation
C:\Users\Master\Desktop\ProjectDrive\M1298\Roadmap.docx
Assigned by : Lautischer - 31.01.2012 20:38
Assigned by : Lautischer - 31.01.2012 20:38

Figure 5.7.: Screenshot GUI - File reviews.

5.1.6. Rights Management

The access to the content of the management system is regulated with access privileges (softgoal *rights management be provided*). There is no need to manually give additional permissions to users because the right management has already been done for the network file system. User groups are introduced with various rights on different folders. If users require additional rights for specific projects, they request permissions from the superior or from the IT department. This simplifies the right management process and secures its uniformity, what furthermore has a positive influence on the softgoal *system security be given*.

Some files only contain data that is not in strict confidence. Those files can be provided for all users of the management system when they are marked properly in the server configuration. After their release they are automatically transferred to a public folder. This accomplishes the softgoal *functionality for public access be provided*. In addition to the existing property information, represented by the user interface, a link to the public folder is added. Users without read access to the project folders are able to follow the new linkage to obtain the needed access. Figure 5.8 shows the representation of this functionality. In this case, "Target Test Specification BE" has been marked in the configuration file.



C:\Users\Master\Desktop\ProjectDrive\M1298\TCAF.docx
Assigned by : Lautischer - 31.01.2012 19:38
Reviewed by: Brandner - 02.02.2012 13:22

C:\Users\Master\Desktop\ProjectDrive\M1298\PRUEFSPECS\BACKEND\PV_BE_KP116_V0.10_2005.06.27.xls
Assigned by : Lautischer - 31.01.2012 20:37
Reviewed by: Brandner - 03.02.2012 09:33

Figure 5.8.: Screenshot GUI - Public folder.

5.2. Unclear Requirements

It is not always possible to entirely fulfill the requirements of all stakeholders. In this project, there are also some requirements that seem to be fulfilled but the completion cannot be totally assured for the whole life cycle of the software. The following "unclear requirements" can be named:

- *system security be given*: the security of the new system is very important for the IT department. Attackers should neither have the possibility to gather confidential information from the management tool nor be able to receive access to harm the server or other internal technologies of the company.

  Many kind of attacks are averted by the IT department because the software runs as an intranet site with certain restrictions. The system is furthermore implemented in a way, where no security gaps arise and confidential data can be protected by permissions on the file system. Nevertheless, so shortly after the introduction of the system it cannot absolutely be ensured that there will never be attacks on the system - even if they are harmless.

- *maintenance effort be minimal*: the maintenance effort for the IT department should be minimal. Therefore, the system does not need any additional effort for keeping the system runnable. These requirements are anyhow marked as "unclear" because in future the need for maintaining the software could appear; for example if the company changes the server-side technology, modifies the data storage on the network file system or introduces new restrictions. In such cases, the IT department may need to adapt the tool or to change the setup to meet the requirements of the new environment.
- *correctness of information be secured*: a correct search index is essential to fulfill the softgoal *search functionality be of high quality*. Provisions that keep the search index correct are already made, but users also need to contribute to a correct index:
  - Project specific information must be stored in the correct project folder, to find it via restricted search requests.
  - If assigned files are deleted, the status of the required file becomes "not assigned" again. New data needs to be assigned to complete the milestone.
  - Revisors must perform revisions seriously to ensure the correctness of the content.
  - Already released files must be reviewed again if changes are made.

6. Conclusion and Outlook

This chapter sums up the results of the performed work in form of a conclusion. The usage of the software in the company and the biggest hurdles during the development process are shown. Finally, ideas for future extensions and improvements are given.

6.1. Contribution

The goal of this thesis was to plan, design, implement and introduce a new software for managing project specific test documents within the Infineon Technologies Austria AG. The requirements engineering process has been performed with the help of the i* Framework, an agent- and goal-oriented approach. According to the i* notation, different models were created to detect the requirements for the management tool and to help on choosing an appropriate technology.

The strategic dependency model gave an overview about all available stakeholders and showed their dependencies. It helped to identify the requirements that needed to be considered to create the software and showed where they came from. To perform further analysis, the rationales behind the dependencies were added to the SD model. The strategic rationale model visualized the reasons for dependencies and their treatment by modeling two potential technologies for the realization of the management tool. A goal evaluation was performed to calculate the labels of goals and softgoals of each actor. This gave an impression about which requirements can be fulfilled if a certain technology is used. This information helped to choose the best-fitting technology where the highest amount of requirements are satisfiable.

The software has been implemented under consideration of internal restrictions. The architecture and the structure of the implementation were visualized and described. The resulted management tool was shown and its functionality was thereby compared with the requirements of the stakeholders. An analysis clarified why the certain parts of the software were created in that special way. Furthermore, it was shown which requirements have been fulfilled and which remained "unclear".

The work at the company showed that in real-life, the requirements engineering process can be as hard or even harder than the technical realization. The development process took eight months and the whole requirements engineering process including the choice of the technology took more than half of this time.

6.1.1. Actual Project Status

At this time, the software is introduced to the company and has already been in common use. It runs stable on the Apache Tomcat Server and is supplied with new data by the crawler every day. All stakeholders are trained and able to use the functionality of the software via the graphical user interface. There is one internal leader for the project who is responsible for managing configuration files and giving a hand on using the system. This person has been involved in the project for the whole development process and has additional technical knowledge as well as privileges for performing restarts of the search index and for causing reloads of the server configuration. In cases of unexpected behavior or problems, the project leader establishes contact to the IT department to ask for further support.

During the first months, project-specific knowledge for all current and already completed projects is transferred to the management tool. The search functionality is able to find project data of all projects immediately. Nevertheless, the project progress and its milestones can only be visualized correctly if the right information is manually assigned by involved parties. Very old projects that were finalized many years ago, are sometimes not really needed any more. In such cases, only the information for the final milestone is assigned by hand for the sake of completeness; previous milestones are left empty.

If new projects are created and processed, the system can be used more efficiently. The entire functionality emerges when the project progress is simultaneously transported to the search index from the beginning of a test project.

6.1.2. Challenges

Some challenges and hurdles that appeared during the development process are listed in this chapter.

- Knowledge exchange: to be able to think about solutions of the problem, it was necessary to understand the internal test processes and circumstances from different points of view. There were several technical terms that complicated this initial training. Before realistic requirements could be determined, knowledge also needed to be transferred to the test engineers. They were given an overview about possible solutions.
- Changes in requirements: although the requirements were well elaborated and planned at the beginning, small changes occurred during the development process. Some stakeholders altered their requirements and completely new needs appeared. Even if these changes were few, their realization cost more time later on.
- IT support: the headquarter of the IT department is stationed in another facility of the company. Thus, there was no in-house contact person for supporting the introduction process of a new software and everything was communicated via the "HelpDesk" intranet page, the communicator and the telephone. It was hard to find the responsible person for a certain problem and solutions were costly elaborated. The development on the Tomcat server was tedious because server restarts due to changes on the software could only be performed by the IT headquarter and also server logs needed to be requested.

6.2. Future Work

The current implementation fulfills the majority of stakeholders' requirements. Nevertheless, there are several improvements and extensions of the document management system and the data storage that could be realized in future:

- Standardization of the folder structure: the actual folder structures of the network drives are not standardized. It is not clear how a consistent naming can be performed and which content belongs to which folder on the project drive. A standard would help to keep the structure of new projects uniform.
- Creation of new projects: additional functionality can be added to the software, that bundles things that need to be done if a new project is created; for example creating a predefined folder structure on the network drive or adding the project to the crawler configuration.
- Extension of the Wiki page: the Wiki page, that acts like a help page for the usage of the management tool, can be extended by a FAQ area or a forum.

- Admin area: an administrative area can be added to the GUI, where the content of the page and the server configuration can be adapted by qualified persons.
- Email notification: the actual process of reviewing files can be improved by an automatic email notification between the assigner and the revisor of information. If users complete their milestones, the supervisor gets a request to review the information. If the review is done, the notification is sent back to the assigner.

A. i* Models

This appendix chapter gives further details of the technology choice. It shows the results of the goal evaluation for the remaining actors.

A.1. Goal Evaluation

Figure 3.11 already showed the evaluation results for the *Active User*. The following figures visualize the evaluation results for the actors *IT*, *Passive User* and *Revisor*. Like in section 3.2.2, differences to the evaluation of technology one are highlighted with colored cycles.

Figure A.1 shows the varying impacts on the requirements of the IT department that are caused by the different technologies. The implementation of the search engine has a negative effect on the maintainability and the security of the system, what is argued in chapter 5.2.

The evaluation of the *Passive User* and the *Revisor* (figures A.2 and A.3) only show improvements for choosing technology two.

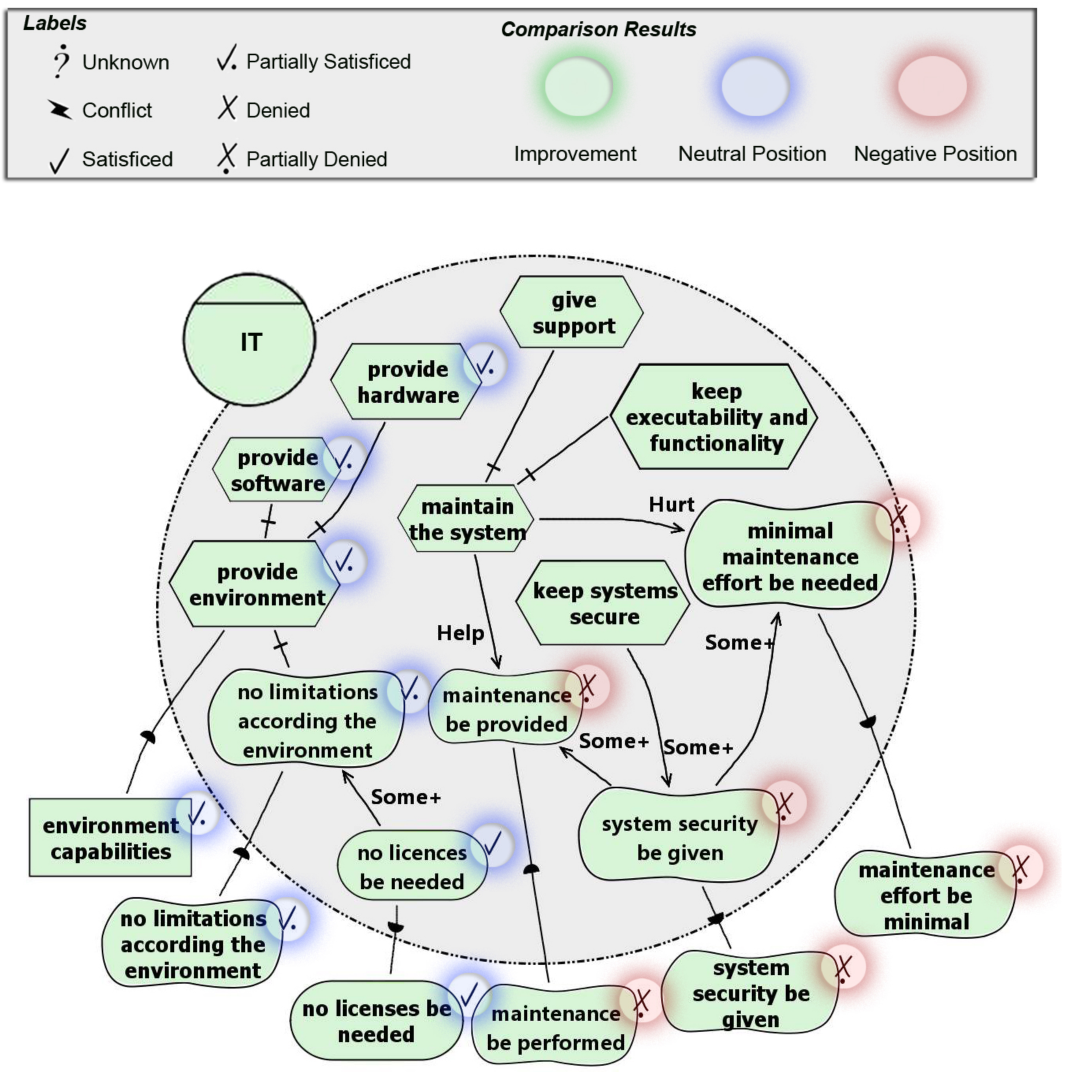


Figure A.1.: Search engine evaluation results - IT.

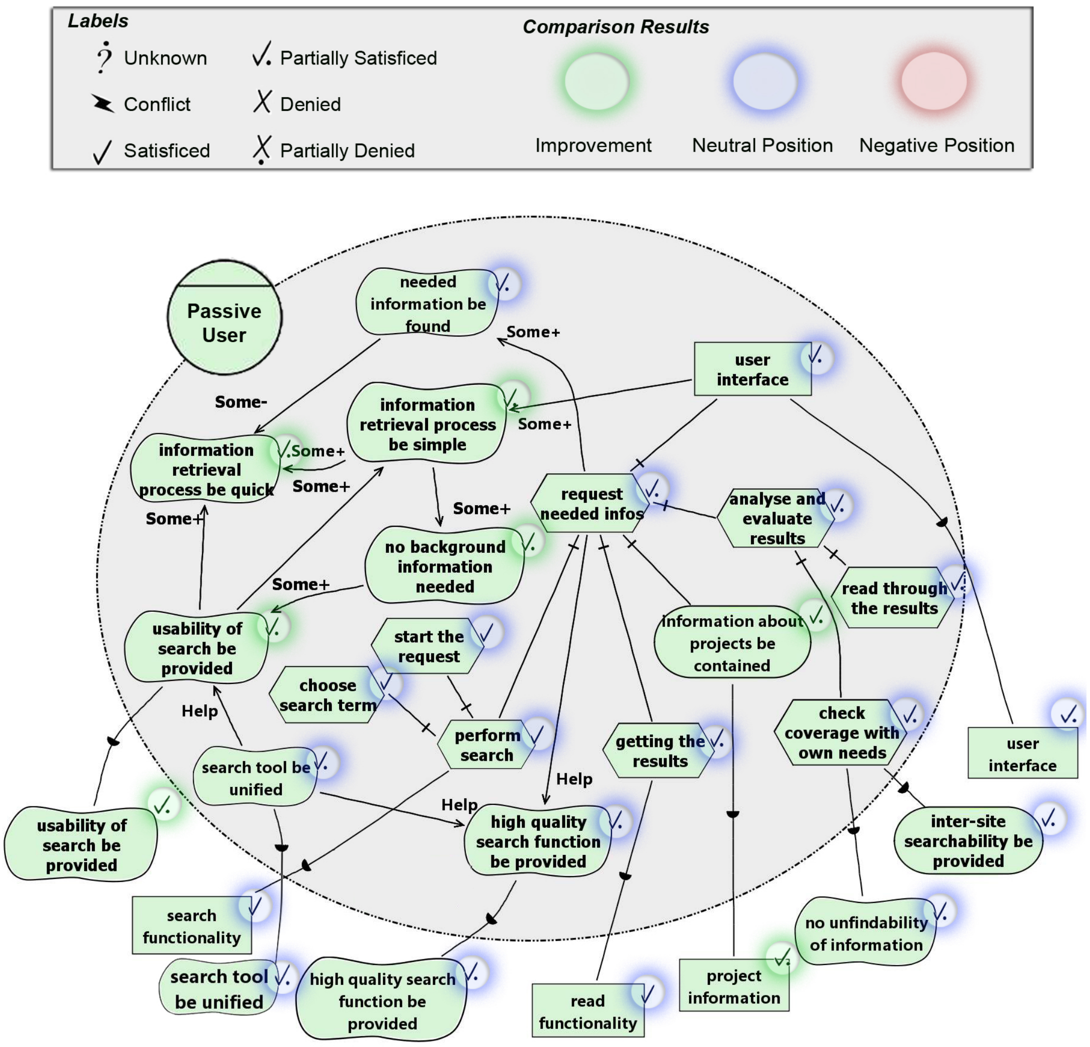

Labels
Unknown
Conflict
Satisficed
Partially Satisficed
Denied
Partially Denied
Comparison Results
Improvement
Neutral Position
Negative Position
Passive User
needed information be found
information retrieval process be simple
information retrieval process be quick
user interface
request needed infos
analyse and evaluate results
no background information needed
read through the results
usability of search be provided
start the request
information about projects be contained
choose search term
perform search
getting the results
check coverage with own needs
user interface
search tool be unified
high quality search function be provided
inter-site searchability be provided
usability of search be provided
search functionality
no unfindability of information
search tool be unified
high quality search function be provided
read functionality
project information
Some+
Some-
Help

Figure A.2.: Search engine evaluation results - Passive User.

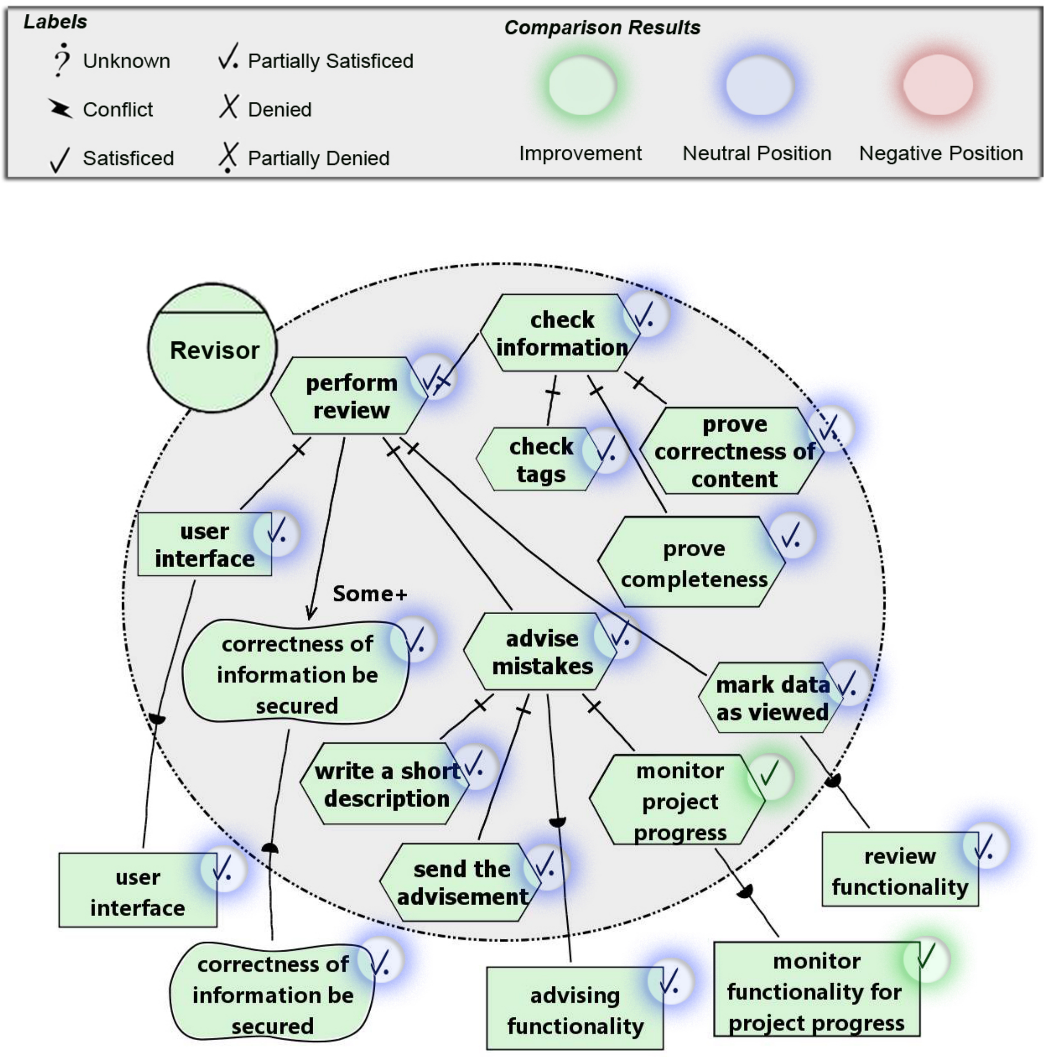


Figure A.3.: Search engine evaluation results - Revisor.

B. Implementation Details

This appendix chapter shows further details of the implementation of the management tool. It mainly focuses on the "embedded version" of Apache Solr, the configuration of the search index and the structure of property files.

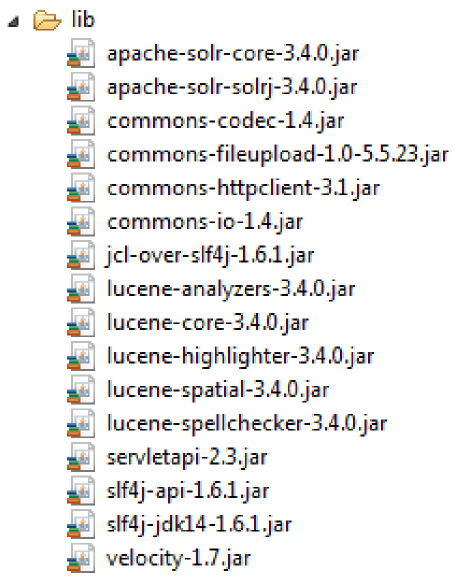
B.1. Embedded Solr

The first implementation of the system used the Apache Solr search index the common way - as a separate web application on the Tomcat server. Therefore, another web application was requested. The GUI communicated via HTTP requests to the servlets, that again performed HTTP requests to the Apache Solr application. This passing of information was time consuming and made the document management system depending on two server applications.

The final implementation uses the "embedded version" of Apache Solr that is provided by the Solrj Java AP. The *ServerConfig* class holds an *EmbeddedSolrServer* object and is responsible for the initialization and the assignment of the home path of the index - the path to the proper folder on the web space of the requested server application. The index can be modified directly from this member object. For each file, an object with the type *SolrInputDocument* is created, where all indicated fields are added in key-value pairs. The new object is then appended to the index by the *add()* functionality of the *EmbeddedSolrServer*. The added information is buffered by the server and taken over after a call of the *commit()* function.

To be able to use the embedded functionality, several external libraries must be included (see figure B.1) which are part of different projects of the Apache Software Foundation. Apache Solr is used in version 3.4 that has been released on the 14 th of September 2011.

If the Solr server does not work properly, it gets restarted automatically by the server application. This ensures steadiness of the document management system and reduces the maintenance effort. Furthermore, the duration of crawler runs is reduced to half of them in the final implementation.



```
lib
    apache-solr-core-3.4.0.jar
    apache-solr-solrj-3.4.0.jar
    commons-codec-1.4.jar
    commons-fileupload-1.0-5.5.23.jar
    commons-httpclient-3.1.jar
    commons-io-1.4.jar
    jcl-over-slf4j-1.6.1.jar
    lucene-analyzers-3.4.0.jar
    lucene-core-3.4.0.jar
    lucene-highlighter-3.4.0.jar
    lucene-spatial-3.4.0.jar
    lucene-spellchecker-3.4.0.jar
    servletapi-2.3.jar
    slf4j-api-1.6.1.jar
    slf4j-jdk14-1.6.1.jar
    velocity-1.7.jar
```

Figure B.1.: Required libraries for embedded Solr.

B.2. Search Index

The search index is configured via the "schema.xml" that is found in the "conf" folder of the Solr distribution. Here, the indicated fields are defined as shown in figure B.2. Values for the first five elements are calculated out of each indicated file and represented by the *FieldEntry* class. Information for the other elements are stored in property files.

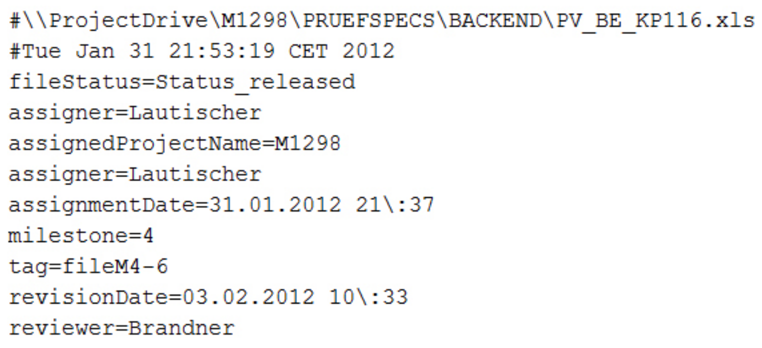
```
<field name="projectName" type="string" indexed="true" stored="true"/>
<field name="fileName" type="string" indexed="true" stored="true"/>
<field name="fileType" type="string" indexed="true" stored="true"/>
<field name="filePath" type="string" indexed="true" stored="true" required="true"/>
<field name="lastModificationDate" type="date" indexed="true" stored="true"

<field name="tag" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="fileStatus" type="string" indexed="true" stored="true" />
<field name="milestone" type="string" indexed="true" stored="true" />
<field name="revisionDate" type="date" indexed="true" stored="true"/>
<field name="reviewer" type="string" indexed="true" stored="true" />
<field name="assignmentDate" type="date" indexed="true" stored="true"/>
<field name="assignedProjectName" type="string" indexed="true" stored="true" />
<field name="assigner" type="string" indexed="true" stored="true" />
```

Figure B.2.: Required libraries for embedded Solr.

B.3. Property Files

Property files are used to store additional file-information that is created by the server application on file assignments and file reviews. The *FileProcessingEngine* servlet uses the *PropertyFile* class to store this additional information in key-value pairs. The absolute path of the corresponding file is used to calculate the path of the property file, that actually lies on the web-space of the server application. With the help of property files, the crawler is able to recognize project specific information like the belonging to a milestone of a certain project. If there are problems with the actual index or the index is even damaged or deleted, there is no reason to panic because it gets restored automatically on the next crawler run.

A property file is deleted by the system if the corresponding file is no longer assigned to a project. Therefore, regular users do not need access to property files and only the internal project leader is familiar of their existence and is able to change them. Figure B.3 shows the structure of a property file for released data.



```
#\\ProjectDrive\M1298\PRUEFSPECS\BACKEND\PV_BE_KP116.xls
#Tue Jan 31 21:53:19 CET 2012
fileStatus=Status_released
assigner=Lautischer
assignedProjectName=M1298
assigner=Lautischer
assignmentDate=31.01.2012 21\:37
milestone=4
tag=fileM4-6
revisionDate=03.02.2012 10\:33
reviewer=Brandner
```

Figure B.3.: Structure of a property file.

Bibliography

[Abdulhadi et al., 2007] Abdulhadi, S., Horkoff, J., Yu, E., and Grau, G. (2007). istarguide. URL:`http://istar.rwth-aachen.de/tiki-index.php?page=i*+Guide` [Last visited on 20.01.2012].

[Becerra-Fernandez and Sabherwal, 2010] Becerra-Fernandez, I. and Sabherwal, R. (2010). *Knowledge management: Systems and Processes.* M.E. Sharpe, Armonk, New York.

[Becker and Huber, 2008] Becker, C. and Huber, D. E. (2008). Die bilanz des (miss)-erfolges in it-projekten - harte fakten und weiche faktoren. URL:`http://d-nb.info/99200375X/34` [Last visited on 10.11.2011].

[Borgida et al., 2009] Borgida, A. T., Chaudhri, V. K., Giorgini, P., and Yu, E. S. (2009). *Conceptual Modeling: Foundations and Applications, Essays in Honor of John Mylopoulos.* Springer, Berlin.

[Brandner, 2011] Brandner, J. (2011). Web-based process documentation.

[Brill et al., 2010] Brill, O., Schneider, K., and Knauss, E. (2010). Videos vs. use cases: Can videos capture more requirements under time pressure? In *Requirements Engineering: Foundation for Software Quality*, pages 30–44. Springer-Verlag. URL:`http://www.springerlink.com/content/p8225h4ku4171474/`.

[Brittain and Darwin, 2007] Brittain, J. and Darwin, I. F. (2007). *Tomcat: The Definitive Guide, 2nd Edition.* O'Reilly, second edition.

[Brooks, 1987] Brooks, F. (1987). No silver bullet essence and accidents of software engineering. *Computer Vol. 20 No. 4*, pages 10–19. URL:`http://people.eecs.ku.edu/~saiedian/Teaching/Sp08/816/Papers/Background-Papers/no-silver-bullet.pdf`.

[Carlson, 2006] Carlson, P. (2006). Apache lucene - query parser syntax. URL:`http://lucene.apache.org/java/2_9_1/queryparsersyntax.pdf`.

[Chung et al., 2000] Chung, L., Nixon, B. A., Yu, E., and Mylopoulos, J. (2000). *Non-Functional Requirements in Software Engineering.* Kluwer Academic Publishers, Boston/Dordrecht/London.

[de Bruijn and Dekkers, 2010] de Bruijn, F. and Dekkers, H. L. (2010). Ambiguity in natural language software requirements: A case study. In *Requirements Engineering: Foundation for Software Quality*, pages 233–247. Springer-Verlag.

[Efendi et al., 2007] Efendi, F., Rowe, S., Grindstaff, D., James, and Hatcher, E. (2007). Embedded solr. URL:`http://wiki.apache.org/solr/EmbeddedSolr` [Last visited on 24.09.2011].

[Foundation, 2007a] Foundation, T. A. S. (2007a). Introduction to the solr enterprise search server. URL:`http://lucene.apache.org/solr/features.html` [Last visited on 27.01.2012].

[Foundation, 2007b] Foundation, T. A. S. (2007b). Solr tutorial. URL:`http://lucene.apache.org/solr/tutorial.html` [Last visited on 27.01.2012].

[Fowler and Scott, 2003] Fowler, M. and Scott, K. (2003). *UML Distilled: A Brief Guide to the Standard Object Modeling Language.* Addison-Wesley, USA.

[Garlan, 1994] Garlan, D. (1994). The role of software architecture in requirements engineering. URL:`http://www.cs.cmu.edu/afs/cs/project/able/ftp/sareqts-re94/sareqts-re94.pdf`.

[Hatcher et al., 2011] Hatcher, E., Sundling, P., and Ryan (2011). Solr wiki. URL:`http://wiki.apache.org/solr/SolrInstall` [Last visited on 14.09.2011].

[Horkoff and Yu, 2009a] Horkoff, J. and Yu, E. (2009a). Evaluating goal achievement in enterprise modeling – an interactive procedure and experiences. URL:`http://www.cs.toronto.edu/pub/eric/PoEM09-JH.pdf` [Last visited on 06.03.2012].

[Horkoff and Yu, 2009b] Horkoff, J. and Yu, E. (2009b). A qualitative, interactive evaluation procedure for goal- and agent-oriented models. URL:`http://www.cs.utoronto.ca/~jenhork/Papers/Horkoff_CAISEForum.pdf` [Last visited on 06.03.2012].

[Horkoff et al., 2006] Horkoff, J., Yu, E., and Grau, G. (2006). istar-quickguide. URL:`http://istar.rwth-aachen.de/tiki-index.php?page=iStarQuickGuide` [Last visited on 24.01.2012].

[IEEE, 1990] IEEE (1990). Ieee standard glossary of software engineering terminology. URL:`http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf`.

[IEEE, 1998a] IEEE (1998a). Ieee guide for developing system requirements specifications (ieee std 1233-1998). URL:`http://asaha.com/download/MNDcxMTg-`.

[IEEE, 1998b] IEEE (1998b). Ieee guide for information technology — system definition — concept of operations (conops) document (ieee std 1362-1998 (r2007)). URL:`http://web.ecs.baylor.edu/faculty/grabow/Fall2011/csi3374/secure/Standards/IEEE1362.pdf`.

[Liu et al., 2003] Liu, L., Yu, E., and Mylopoulos, J. (2003). Security and privacy requirements analysis within a social setting. *Proc. of the 11th International Requirements Engineering Conference (RE'03)*, pages 151–161. URL:`http://www.cs.toronto.edu/pub/eric/RE03.pdf`.

[Mylopoulos et al., 1990] Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M. (1990). Telos: representing knowledge about information systems. *ACM Transactions on Information Systems Vol. 8 No. 4*, pages 325–362. URL:`http://doi.acm.org/10.1145/102675.102676`.

[Partsch, 2010] Partsch, H. (2010). *Requirements-Engineering systematisch: Modellbildung fuer softwaregestuetzte Systeme*. Springer, Berlin, Heidelberg.

[Ryan et al., 2011] Ryan, Noble, P., Mangar, S., and Szott, S. (2011). Solrj. URL:`http://wiki.apache.org/solr/Solrj` [Last visited on 25.09.2011].

[Schwartz, 2006] Schwartz, D. G. (2006). *Encyclopedia of Knowledge Management*. Idea Group Reference.

[Strohmaier et al., 2007] Strohmaier, M., Aranda, J., Yu, E., Horkoff, J., and Easterbrook, S. (2007). Analyzing knowledge transfer effectiveness - an agent-oriented modeling approach. In *HICSS*, page 188. IEEE Computer Society. URL:`http://kmi.tugraz.at/staff/markus/documents/2007_HICSS_Knowledge-Transfer.pdf`.

[Strohmaier et al., 2008] Strohmaier, M., Horkoff, J., Yu, E., Aranda, J., and Easterbrook, S. (2008). Can patterns improve i* modeling? two exploratory studies. In *REFSQ '08: Proceedings of the 14th international conference on Requirements Engineering*, pages 153–167, Berlin, Heidelberg. Springer-Verlag. URL:`http://kmi.tugraz.at/staff/markus/documents/2008_REFSQ08_Patterns.pdf`.

[Wieringa and Persson, 2010] Wieringa, R. and Persson, A. (2010). *Requirements Engineering: Foundation for Software Quality: 16th International Working Conference, REFSQ 2010*. Springer, Berlin, Heidelberg.

[Yu, 1995] Yu, E. (1995). *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada. URL:`ftp://ftp.db.toronto.edu/pub/eric/DKBS-TR-94-6.pdf`.

[Yu, 2011] Yu, E. (2011). i* an agent- and goal-oriented modelling framework. URL:`http://www.cs.toronto.edu/km/istar/` [Last visited on 30.01.2012].

[Yu et al., 1998] Yu, E., Bois, P. D., Dubois, E., and Mylopoulos, J. (1998). From organization models to system requirements - a "cooperating agents" approach. URL:`http://www.cs.toronto.edu/pub/eric/CoopIS95.pdf`.

[Yu et al., 2011] Yu, E., Giorgini, P., Maiden, N., and Mylopoulos (2011). *Social Modeling for Requirements Engineering*. The MIT Press.

[Yu and Liu, 2000] Yu, E. and Liu, L. (2000). Modelling trust in the i* strategic actors framework. In *Proc. of the 3rd Workshop on Deception, Fraud and Trust in Agent Societies at Agents2000*, pages 3–4. Springer-Verlag. URL:`ftp://ftp.cs.utoronto.ca/pub/eric/Trust00.ps.gz`.

[Yu et al., 2001] Yu, E., Liu, L., and Li, Y. (2001). Modelling strategic actor relationships to support intellectual property management. In *Proceedings of the 20th International Conference on Conceptual Modeling: Conceptual Modeling*, pages 164–178, London, UK, UK. Springer-Verlag. URL:`ftp://ftp.cs.utoronto.ca/pub/eric/ER01-sub.pdf`.

[Yu, 2009] Yu, E. S. (2009). Social modeling and i*. In *Conceptual Modeling: Foundations and Applications*, pages 99–121, Berlin, Heidelberg. Springer-Verlag.