LUND UNIVERSITY
School of Economics and Management

A Study on Software Requirements Specifications

- some examples

Bachelor thesis, 15 credits, INFK01 in informatics

Presented: January, 2010

Author: Tanja Rolandsson

Supervisor: Björn Johansson
Examiners: Agneta Olerup, Erik Wallin

Abstract

Title: A Study on Software Requirements Specifications - some examples

Author: Tanja Rolandsson

Publisher: Department of Informatics, School of Economics and Management, Lund University

Supervisor: Björn Johansson

Examiners: Agneta Olerup, Erik Wallin

Year of publication: 2010

Type of thesis: Bachelor thesis

Language: English

Keywords: Software requirements specification, requirements engineering, functional requirements, non-functional requirements, IEEE 830

The software requirements specification (SRS) is an important document, since it forms a basis for subsequent activities in the systems development process. In this thesis, nine SRSs are analyzed in order to demonstrate similarities and differences in SRS composition and requirements organization, and to show what type of requirements is the most common one. The results show that SRSs are structured either by following the IEEE (Institute of Electrical and Electronics Engineers) standard 830 with three main sections (introduction – overview – list of requirements), or another structure (introduction – references – list of requirements). How the specific requirements in the SRSs are structured differ from SRS to SRS. The most frequent type of requirements is functional requirements. This shows that even though using standards might not be the only way to formulate documents, they are being used and serve their purposes, at least to some extent.

Table of contents

1. Introduction

The overall systems development process is usually considered to consist of a number of steps, and one of those steps is the system requirements determination (SRD). SRD is a process which in itself includes several different activities, and can be defined as "the overall process of getting at, analyzing, and documenting the requirements" (Duggan & Thachenkary, 2003). It is an important activity in the systems development process, since it forms a basis for subsequent activities, affects the design of the system architecture and contributes to the quality of the system (Hull et al, 2005; Wiktorin, 2003). Hull et al (2005) describes several reasons for project failure, and incomplete requirements are one of the major reasons for failed projects. However, even if projects succeed, insufficient requirements can have several unwanted consequences, such as lower systems quality, unsatisfied users or the development taking longer time and becoming more expensive than expected (Eriksson, 2007). Another benefit of well-formulated requirements is that they can be one of the most important reasons for project success (Hull et al, 2005; Wiktorin, 2003). Apparently, the SRD is of great value to the systems development process, and adequate requirements are a necessity for project success.

There are a number of methods to collect requirements, and once the requirements have been collected, they need to be documented. This documentation usually results in a requirements specification. As part of the SRD, the software requirements specification (SRS) is an important document in the development project. The SRS can be a channel of communication, conveying the characteristics of the system between developers and users. It is also often a part of a contract and can be used to evaluate the systems performance. Moreover, it can be used to estimate time and cost for the development project (Daniels & Bahill, 2004; Smith et al, 2007). It is not always obvious how the SRS should be formulated, but there are templates available. For example, he Institute for Electrical and Electronic Engineers (IEEE) has constructed a standard called 830 which lines up a number of attributes the SRS should have in order to be a well-formulated, understandable document (IEEE, 1998a). This standard is widely mentioned in the requirements engineering literature (e.g. Eriksson, 2007; Smith et al, 2007; Wiegers, 1999; Wiktorin, 2003). However, Power (2002) researches requirements documentation in her thesis, and finds that in contrast to what is described in literature, actual practitioners of requirements documentation use a variety of methods, both stylistically and structure-wise, when documenting requirements. In addition, Smith et al (2007) write that "[…] there is no universally accepted way of documenting requirements" (p. 88).

Literature on requirements determination (e.g. Avison & Fitzgerald, 2006; Cysneiros & Leite, 2004) claims that one certain type of requirements, more explicitly functional requirements, are specified more clearly and to a larger extent than other, nonfunctional requirements, in requirements specifications. The explanation for this is said to be that nonfunctional requirements are more difficult to identify than functional requirements. This in combination with the information given above raises a couple of questions which this thesis seeks to answer:

- How are software requirements specifications structured and the requirements in them organized?
- What is the most common category of requirements in actual software requirements specifications?

The first research question for this thesis is thus to find out how existent SRSs are structured – if they follow standards described in literature on requirements engineering, such as the IEEE 830, or if the structure of SRSs and the arrangement of the requirements follow more personalized standards, as was the result of Power's (2002) research. The second research question is to examine whether a certain type of requirements is more common than others or not and, if this is the case, what type that is. These questions will lead to the purpose of the thesis, which is to identify similarities and differences in SRS composition and requirements organization, illustrate what types of requirements are most frequent, and show how requirements are formulated.

In order to answer the above questions, a number of SRSs will be analyzed. They are finished SRSs for different software products, available on the internet and written in the last decade (year 2000 or later). These SRSs will be analyzed with a deductive-logic approach, which means that it is an analysis of whether a phenomenon differs from earlier created theories or not (Halvorsen, 1992). In practice, this means that there will be a theoretical background used to compare the SRSs to a) this background and b) each other. Further limitations and criteria for selection of data are discussed in chapter three.

Requirements specifications have been considered in a number of earlier essays at different universities. Besides the research by Power (2002) mentioned above, Franko & Hansson (2006) examine the impact of organizational rules when formulating a requirements specification and Dahlstrand et al (2009) suggest a framework for developing a requirements specification when using a particular systems development technique called Co-design. These last two, however, focus on how to create a well-formulated SRS. This thesis deals with the outcome of such efforts.

The next chapter presents a selection of literature on SRSs and requirements in order to create the theoretical background for the analysis. It shows different ways SRSs can be structured and how requirements can be categorized, and presents an outline for analysis. Method is discussed in chapter three, to explain how data was collected and how the analysis will be conducted. After this, the results from each analyzed SRS are presented, followed by an analysis and discussion of the results. Finally, there are some concluding remarks.

2. About specifications and requirements

This chapter contains the information necessary to build a theoretical background upon which the analysis will be based. Firstly, requirements specifications are defined and explained, and a standard for structuring requirements specification is presented. Secondly, there is a paragraph on formal/informal language. Thirdly, a categorization of requirements and a number of different types of requirements are described, and finally a description of the baseline for the analysis will be presented.

2.1 Fundamentals of software requirements specifications

As mentioned in the introduction, the software requirements specification (SRS) is an important document for the systems development process. It is used by different groups of people for different purposes; by customers, to know what to expect, by the software developers, to know what to build and how, by test groups, to test and evaluate the system and so forth (Hull et al, 2005; Wiegers, 1999). The SRS can work as a channel of communication between developers and customers and help to ensure that the system satisfies customer needs. Moreover, it creates the baseline upon which following systems development activities are based. It is obvious that every system has requirements, and the SRS exists to make these requirements possible to build (Daniels & Bahill, 2004; Hull et al, 2005).

However, this does not explain what an SRS is. There are different ways to formulate a definition of the term, and many writers and researchers make their own definition of it. It is quite obvious that the SRS is one part of the overall systems requirements determination process which in its turn is part of the entire systems development process, but it might be uncertain what an SRS actually is, why it is necessary, and what information it is supposed to contain.

Eriksson (2007) describes the SRS as a document which is usually produced when a system is built from scratch or if there are major changes being made to an existing system. That might be useful information, but it is a very brief description. Wiktorin (2003) writes that "a requirements specification consists of several parts." (p. 40[1]). That sounds rather unclear and is not of help when trying to understand the concept of requirements specifications. It does reveal though that there needs to be more than one activity in creating the document. Another description, also rather short, is given by Duggan & Thachenkary (2003): "Requirements specification: representing the results [of the previous steps in the SRD process] in a document" (p. 392). This explains where an SRS comes from, but not what information lies within it.

One explanation of the SRS and its contents is given by Wieringa (1996), who states the following: "A requirements specification consists of a specification of product objectives and a specification of required product behavior" (p. 71). In other words, the SRS shows the purpose of the system, and how it is supposed to behave – its functionality. Since the SRS is important for the following activities in the systems development process, it needs to have a detailed description of system behavior (Wiegers, 1999). Smith et al (2007) use a similar definition as the one given by Wieringa; they state that the SRS should describe essential

[1] Translated from Swedish: "En kravspecifikation består av flera delar."

system requirements of the software and its external interfaces, such as functions, performance, constraints and quality attributes. Another similar description of the SRS is given by IEEE in the standard 12207: "the systems requirements specification shall describe: functions and capabilities of the system; business, organizational and user requirements; safety, security, human-factors engineering (ergonomics), interface, operations, and maintenance requirements; design constraints and qualification requirements" (IEEE 1998b, p 17). To sum up, the SRS is a document created when a system is built or rebuilt. It contains the purpose and behavior of the system; descriptions of the system and its desired functions.

2.1.1 Recommended contents of a requirements specification

The IEEE (Institute of Electrical and Electronics Engineers) standard 830 called Recommended Practice for Software Requirements Specification is a standard where an outline for a requirements specification structure is given (IEEE, 1998a; Wiktorin, 2003, p. 122). IEEE 830 is also mentioned recurrently in other literary works (e.g. Eriksson, 2007; Smith et al, 2007; Wiegers, 1999; Wiktorin, 2003). According to this standard, a requirements specification should contain three main sections; introduction, overview and a list of requirements.

Wiegers (1999) presents a modified version of the outline given in the IEEE standard 830 (p. 154). His template extends the list of requirements into a) external interface requirements b) system features c) other non-functional requirements and finally d) other requirements. Paragraphs 1), introduction and 2), overview are, with a couple of minor modifications, the same as presented by Wiktorin (2003). The table below shows the outlines for IEEE 830 and the extended version.

There are most likely a large number of other templates available as well, but since the IEEE 830 has been frequently mentioned in the literature used in this thesis, it will be the basis for the analysis.

Wiktorin (2003) suggests that it is of help if the requirements in the list of requirements are organized in different groups, as Wiegers (1999) shows in the template described above. Wiktorin (2003) also states that since functional requirements usually are numerous, it can be necessary to have several sub-categories of functional requirements, to make the interpretation and understanding of the requirements easier. Wiegers (1999) points out that it is necessary to give each requirement a unique identifier. The simplest way to do this is to use a sequence number which shows both type of requirement and number. Another way is to use hierarchical numbering, which is claimed to be the most common way to label requirements (Wiegers, 1999).

Table 2.1. IEEE 830 (IEEE, 1998a) and its extended version by Wiegers (1999).

IEEE 830	Modified IEEE 830 according to Wiegers (1999)
1. Introduction a. Purpose b. Scope c. Definitions, acronyms, abbreviations d. References e. Overview	1. Introduction a. Purpose b. Document convention c. Intended audience and reading suggestions d. Product scope e. References
2. Overall description a. Product perspective b. Product functions c. User characteristics d. Constraints e. Assumptions and dependencies	2. Overall description a. Product perspective b. Product functions c. User classes and characteristics d. Operating environment e. Design and implementation constraints f. Assumptions and dependencies
3. Specific requirements Appendices Index	3. External interface requirements a. User interfaces b. Hardware interfaces c. Software interfaces d. Communications interfaces 4. System features e. System feature X i. Description and priority ii. Stimulus/response sequences iii. Functional requirements 5. Other nonfunctional requirements f. Performance requirements g. Safety requirements h. Security requirements i. Software quality attributes j. Business rules k. User documentation 6. Other requirements Appendix A: Glossary Appendix B: Analysis Models Appendix C: To-Be-Determined List

2.2 Language

The requirements can be documented in different ways from a language point of view. There are formal methods, where mathematically formal syntax and semantics is used to describe the requirements, and informal methods, where the requirements are described in natural language (Smith et al, 2007). An example of a formal language is the Vienna Development Method Specification Language (VDM-SL). SRSs where such a language has been used to formulate requirements can have statements as the one in the figure below:

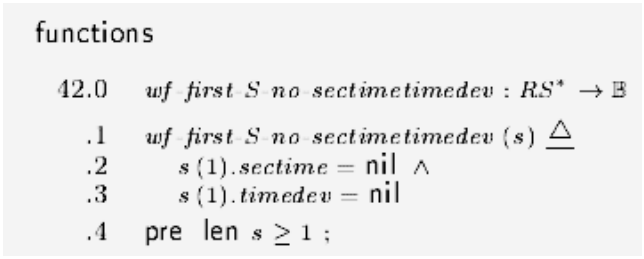

Figure 2.1. Example of VDM-SL in use (Droschl, 2000, p. 19).

The latter approach, with natural language, is arguably the most common one in contemporary SRSs (Nicolás & Toval, 2009). Wiegers (1999) seems to assume that requirements will be written in natural language, and gives a number of guidelines for writing requirements, such as "state requirements in a consistent fashion […]" (p. 162) and "avoid comparative words […]" (p. 163). The level of formality in the SRS does not only depend on what the analyst wants to write. It also depends on the complexity of the system. Highly complex systems require a higher level of formality (Daniels & Bahill, 2004).

2.3 Categorizing requirements

As stated above, the readability of the SRS can be improved by categorizing the requirements in some kind of way. Requirements can have very varying characteristics, so categorizing them by dividing them into different types would be a logic way. But first, it is necessary to define what a requirement is.

Machado et al (2005) presents the following different ways to define the term requirement:

> "(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)." (Machado et al, 2005, p. 47)

The first and second of these definitions are applicable to the requirements in SRSs, and are hereafter used in this thesis. The third definition appears to be what in this thesis is considered an SRS, and does not apply to the term requirement in this context.

Requirements are, as mentioned earlier, not all of one kind, and there are many ways to classify them. Eriksson (2007) divides requirements into three broad categories, where the second category has a number of sub-categories. These different types of requirement categories are functional, non-functional (with the sub-categories usability, reliability, performance & supportability) and design restrictions. Wiktorin (2003) divides requirements into functional and non-functional ones. He also adds sub-categories described in an ISO (International Organization for Standardization) standard (ISO 9126) to further categorize requirements. The first sub-category from ISO 9126 is function, which fits under functional requirements, and the rest of the ISO 9126 categories (reliability, usability, efficiency, maintainability and portability) are considered non-functional aspects (Wiktorin, 2003). Avison & Fitzgerald (2006) describe a quite similar division: on one hand there are functional requirements, on the other hand non-functional requirements with a number of sub-categories. Another categorization is made by Grady (1992), who describes the FURPS+ model,

originally used by Hewlett-Packard to help their developers organize customer needs. FURPS is an acronym for Functionality, Usability, Reliability, Performance, Supportability (Grady, 1992). If these sound familiar it is because they are the same ones used by Eriksson (2007). The "+" sign was added to the model to "extend the acronym to emphasize various specific attributes" (Grady, 1992, p. 32). Yet another division is presented in the IEEE standard 830 (IEEE 1998a). This standard suggests the following groups of requirements: external interfaces, functions, performance requirements, logical database requirements, design constraints and software system attributes. The last category has five sub-categories; reliability, availability, security, maintainability and portability (IEEE, 1998a). These names are fairly similar to the sub-categories of Eriksson's (2007) non-functional requirements, thus does the "software system attributes"-category of the IEEE standard 830 seem to be quite coherent with the non-functional category shown by Eriksson (2007).

This thesis will henceforth use the same categorization as Eriksson (2007) to describe different types of requirements, since it covers the other categorizations well and gives a useful structure for the following sections in this chapter. One modification has been done to this structure though; Eriksson (2007) also divides supportability into maintainability and testability, this division is left out here. Figure 1 below shows the hierarchy of requirements.

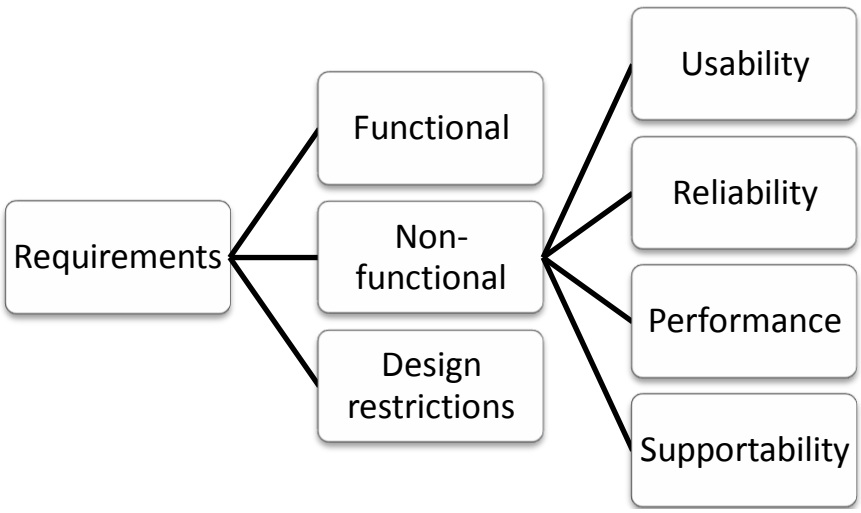

Figure 2.2: Hierarchy of requirements adopted from Eriksson (2007, p. 41), sub-categories of supportability deleted.

2.3.1 Functional requirements

When people think of requirements, functional requirements are usually what they have in mind. Requirements of this type are often descriptions of what the system is supposed to do – the specific functions it is supposed to have (Eriksson, 2007). Another way to express this is that functional requirements describe functions the system needs to have in order to help the end-user with his/her tasks (Wiktorin, 2003). Avison & Fitzgerald (2006) state that functional requirements can be identified by looking at how they are formulated – "anything that is phrased in a noun-verb form is a functional requirement" (p. 106). The FURPS+ keywords for functionality include words as *capability*, *generality* and *security* (Grady, 1992).

2.3.2 Non-functional requirements

If functional requirements answer the question *what* the system is supposed to do, non-functional requirements answer the question *how* the system is supposed to work. Eriksson (2007) decomposes non-functional requirements into four sub-categories, which should be enough for most systems development projects. He states, however, that these categories are not set in stone and that there are a myriad of different ways to sub-categorize non-functional requirements. The four categories are *usability*, *reliability*, *performance* and *supportability* (Eriksson, 2007).

Non-functional requirements can be difficult to identify and they are sometimes forgotten since they have so many aspects to them and because there are difficulties evaluating and measuring them (Avison & Fitzgerald, 2006; Wiktorin, 2003). To identify non-functional requirements, Avison & Fitzgerald (2006) do again argue that looking at how the requirements are formulated can help, stating that adverbs or adverbial modifying clauses are used to express non-functional requirements. Even though non-functional requirements tend to be neglected, they are still important. As Wiegers (1999) states: "[…] they make the difference between a product that merely does what it is supposed to do and one that delights its customers" (p. 196). To describe non-functional requirements in a clearer way, more detailed descriptions of the four categories of non-functional requirements (usability, reliability, performance and supportability) will follow.

Usability is related to how easy it is to learn and use the system (Eriksson, 2007). This includes human factors, aesthetics, consistency and documentation, according to FURPS+ (Grady, 1992). It can be statements regarding what the graphical interface is supposed to look like or that the user is not obliged to use the mouse. Sometimes the terms "ease of use" or "human engineering" are used to refer to usability (Wiegers, 1999). General usability aspects are not necessarily described in relation to every single requirement in the specification, but can be documented in an independent document (Eriksson, 2007).

FURPS+ keywords in the category *reliability* are for example frequency of failure, recoverability and accuracy (Grady, 1992). Eriksson (2007) states that reliability is often expressed by failure frequency, sometimes combined with level of severity of the failure. A failure of low severity can be more acceptable than a failure of high severity. Other aspects to reliability can be measuring the number of correctly performed operations or how often a defect is identified in the system (Wiegers, 1999). Like usability requests, general requirements of reliability can be stated in an individual document called an SLA (Service Level Agreement) (Eriksson, 2007).

Performance requirements are usually related to time, such as response time, recovery time, or number of transactions per time unit. They are supposed to make sure the user does not have to wait for results to show up on the screen and help avoid system overload (Eriksson, 2007; Wiegers 1999). They are mostly specified quantitatively which makes them easily measured, but they can also be specified qualitatively (Ho et al, 2007). FURPS+ keywords for performance include speed, efficiency and throughput (Grady, 1992).

Systems need to be easy to manage and maintain, and *supportability* requirements help state these needs. This type of requirements is also helpful to determine how the system can

support troubleshooting and testing, or how easily the system can be changed (Eriksson, 2007; Wiegers, 1999). In FURPS+ there are a lot of words related to supportability, and a few of them are testability, maintainability, configurability and serviceability (Grady, 1992).

2.3.3 Design restrictions

Design restrictions determine things such as what programming language should be used, how the development process is supposed to be documented or if the system has to be developed in a certain environment (Eriksson, 2007). Such restrictions are sometimes considered as a type of non-functional requirement (Avison & Fitzgerald, 2006). Design restrictions can be necessary, especially if the system has to work in a certain environment or communicate with other systems (Eriksson, 2007).

2.4 Outline for analysis and summary

In chapter one, two main research questions were raised. The first one was to examine how SRSs are structured and in what way the specific requirements are organized. The second one was to see if some type of requirements is more common than other types, and in that case, what type that is.

To answer the first question properly, it is necessary to go back to the fundamentals of SRSs. I wrote that the SRS shows the purpose of the system and its functionality. That information is useful when determining if the documents I collect are actual SRSs or something else. Then there was the outline of the three SRS templates. Do the SRSs follow the IEEE 830? If not, how does the structure differ from IEEE 830? Is the language formal or informal? The third paragraph in IEEE 830, list of requirements, can be organized in different ways. This is the most interesting part of question number one. Are the requirements categorized by type, importance, or some other way? Are they given unique identifiers?

This leads to question number two. According to the literature (e.g. Avison & Fitzgerald, 2006; Cysneiros & Leite, 2004), functional requirements are more common than non-functional requirements. Is this consistent with reality? To answer that question, it is first necessary to define what a functional and non-functional requirement is. This has been done in previous sections of this chapter. The list of requirements in each SRS, assuming there is one, will be analyzed, different types of requirements identified and this will give an answer to if the statement that certain types of requirements are more common than others is true or not. In order to answer these questions, the outline for analysis presented in the table below was constructed.

Table 2.2. outline for analysis

Structure of SRS:	<IEEE 830/other>
Language:	<natural/constructed>
Categorization of requirements:	<description of categorization>
Identification of requirements:	<type of identifier>
Number of different categories of requirements:	Functional: Usability: Reliability: Performance: Supportability: Design restrictions:

To summarize this chapter, it started out with explaining what an SRS is, stating that it is a document which is written when developing an information system or making major changes to one. The SRS describes the purpose and functionality of the system. After this, it was stated that the SRS needs to be structured in some way to simplify the readability, and the IEEE standard 830 was presented as an example, along with an extended version. The matter of formal or informal language was then concerned, followed by categorization of requirements. Finally, the outline for analysis was presented in a table.

3. Method

There are a number of important issues to consider when doing research. This chapter deals with different issues related to method. To begin with, the selection of data is discussed, followed by choice of data. After this, there is a section on analyzing data, another one on ethical issues and finally a section on validity and reliability.

3.1 Selecting and collecting data

Data for empirical research can be of different types, such as observations, interviews or documents (Svenning, 2003). In this thesis, the collected data is a number of requirements specifications available online. The selected requirements specifications were found by using a regular Google search. They are requirements specifications for software products of different types, and are finished documents. Keywords used were "software requirements specification" and "system requirements specification". These terms are not very specific, and result in extremely many hits. Instead of trying to limit the search to get a manageable number of hits, the first 25 pages with hits were skimmed and sites which seemed relevant were opened. Google sorts hits by relevance, so the first pages ought to be the ones most likely to hold useful documents. This way, I ended up with 16 possible candidates. After looking through these documents, ten documents were selected. The six ones not selected turned out to be either school projects, too old or not requirements specifications for software.

Selecting data is critical for research, but it can be slightly difficult. There are a number of different strategies available for collecting the right type of data, for example probability sampling, convenience sampling and purposeful selection (Maxwell, 2005). In this thesis, the data has been selected with a purposeful approach. This approach means that the selected material has been deliberately chosen to provide the right information. The benefit of using purposeful selection is to find a representative sample for the research question, something probability sampling can achieve only if the collected data is of large amount. Another benefit is that purposeful selection ensures that the theoretical background is applicable (Maxwell, 2005). This does not mean that the examples are handpicked to give a particular result; it only means they were able to provide the information necessary to answer the research questions.

Choosing data available on the Internet can be risky and in order to get reliable data it is necessary to keep a critical eye when collecting it. One of the major principles in source criticism is to understand *who* the author of something is (Svenning, 2003). If there is no author, the data is of little reliability. Svenning (2003) also claims that the second major principle in source criticism is the origin of the data; *where* does it come from? To avoid getting unreliable data, only requirements specifications which measured up to a certain standard were collected. These criteria were the following three: they needed an origin, they had to be authentic (i.e. no mock-ups), and they had to be written in the last decade (year 2000 or later).

Each requirements specification needed to have a traceable origin, or at least a possibility to find out who the owner was or who had created it. To further establish the reliability and to ensure that it was okay to use the specification in this thesis, an e-mail was sent to the given or probable author or owner of the document. Only two people answered my e-mail, and gave me permission to use their documents. The remaining eight did not answer. Two SRSs had an

explicit prohibition to use it stated in the document. Of these two, one author gave permission to use the document. The other author did not answer my e-mail; therefore this SRS was deleted from the collection. Since there were no limitations printed in the rest of the documents I decided to use them. Finally, I ended up with nine SRSs.

The second criterion was related to authenticity. The requirements specifications had to be from software development projects, no school projects were collected. It was easy to identify school projects; they usually had course name, student name and teacher name stated on the first page. The final criterion, that they needed to be written no earlier than year 2000, was easy to check since almost all SRSs found had a date and version number on them. Undated documents were rejected.

3.2 Analyzing data

Analyzing qualitative data is an interpretive activity and because of its close relation to human beings, it is susceptible to attitudes and hues. This makes it imprecise, but it is nevertheless contributing to science since it can open up new perspectives on situations or give suggestions to further research (Svenning, 2003). The interpretation of the data will be influenced by the researcher's previous understandings of the world and the events in it. In hermeneutics, this is phenomena is called pre-understanding. This understanding will develop in different ways during the research process (Svenning, 2003). Another author might interpret the SRSs in this thesis differently, but with the support of the theoretical background, any major misinterpretations will hopefully be avoided.

Analysis is generally based on finding patterns and one of the keywords in qualitative analysis is sorting. The collected data needs to be sorted in order to be able to analyze it. This is of great importance when analyzing texts (Halvorsen, 1992; Svenning, 2003). Another point Svenning (2003) makes regarding analyzing texts, is that it is crucial to have a theoretical basis for the analysis. This is probably originally meant to apply to literature, such as newspapers or fiction, but there is no obvious reason why it would not apply to other text documents as well, such as requirements specifications, which is the case in this thesis.

What is important to remember when researching phenomena, especially in social sciences, is that there is no absolute truth and that there will be exceptions and cases not following the theoretical rules. On very few occasions can any general conclusions be drawn, the results will most likely be hypothetical (Svenning, 2003).

I do not expect my results to be the ultimate truth, because the research questions and the entire research process are influenced by my personal perspective. Also, since the selected documents are easily available, they might not be representative for all SRSs. There might be other SRSs held secret which are structured in completely different ways. But hopefully it can extend my and maybe other peoples' understanding of the concept of requirements specifications, or give opportunities for them to make other interpretations of the subject. SRSs usually follow a certain structure, as stated in the theory chapter. This will most likely simplify the analysis compared to studies of literature, where it is necessary to actively search for themes and find ways to sort the data.

3.3 Ethics

As in real life, ethical issues can be raised in most scientific areas. This is primarily in question when using humans as the object for research, but there are a number of ethical issues of more general character that apply in other cases as well. Four of these issues will be considered in this section.

Halvorsen (1992) states that the results of research should strive to have good compliance with reality. This means that any claims or conclusions documented should be controllable. They need to be justified and connected to reliable sources available for other researchers. Truth and honesty are guidelines for all scientific research. However, the ethical principle of protecting informants and sources might be in conflict with this (Halvorsen, 1992). In this thesis, there are not really any actual humans involved. This makes it easy to follow the first guideline. All documents are as stated above available online for anyone to analyze them.

The second and third issues are related to the selection and usage of data. In order to get trustworthy results, data needs to be collected systematically. The way data is handled should also be as careful as possible to make sure the research is done in a reliable way (Halvorsen, 1992). To ensure this, the data was systematically collected as described above and the theoretical background was thoroughly built up before analyzing the collected documents.

Halvorsen's (1992) fourth ethical issue has to do with results – they should be presented in a way that allows control and criticism. He further writes that this means that if all the above issues are to be taken care of, other researcher need to be able to use the same primary data. That is only possible in an ideal world, but researchers should seek to meet this as far as possible (Halvorsen, 1992).

3.4 Validity and reliability

Validity is related to how well the research can be used to measure what it was set out to measure. Validity can be divided into internal and external validity. Internal validity has to do with smaller parts of the project, such as the connection between theory and data, or that the right instruments have been used. External validity regards the project in a larger context – is it possible to draw general conclusions from this research? In order to achieve high validity, it is necessary to collect the right type of data (Halvorsen, 1992; Svenning, 2003). The theoretical background of any research will probably be criticized, which creates a need to discuss and justify the choice of theory and data. To achieve high validity in qualitative research, it is also important to discuss alternative interpretations of the results (Maxwell, 2005; Svenning, 2003). There is probably a myriad of other sources I could have used for the theoretical background, however, the ones I found were relatively easy to assemble and appeared reliable. The possibility to draw any general conclusions from the results in this thesis is most likely very limited. Other sources for the theoretical background and other data might give completely different results. Even so, the results here can be seen as useful examples of a phenomenon.

Reliability is another way to make sure the research is trustworthy and of good quality. In a very basic form, it usually means that similar surveys under similar conditions should give similar results. Reliability is crucial for research, without it no data could be used to test

hypotheses (Halvorsen, 1992; Svenning, 2003). This can be difficult to achieve, since qualitative analysis is a highly interpretive activity and can, as mentioned above, be affected by the researcher's attitude or conception of the world. Reliability is more important when doing quantitative research (Svenning, 2003).

3.5 Summary

This thesis started out with raising two major research questions. The first question was to examine how SRSs are structured in order to show similarities and differences in SRS composition, and the second question was to find out which one is the most common requirement type.

These questions are to be answered in a deductive manner, where a theoretical background is the basis for the analysis of data. The empirical data consists of nine SRSs, available on the internet. They were selected with a purposeful approach in order to get the right type of documents and in order to be selected they had to measure up to a particular standard.

Because of the limited number of SRSs used in this thesis, it is impossible to draw any general conclusions and reach high validity. The same goes for reliability – nine other SRSs might give completely different answers, and another author might categorize the requirements differently. Still, there is no need to dismiss the results from this analysis; they should be seen as interesting examples from the real world.

4. Presentation of the results

This chapter presents the results from each SRS one by one, in alphabetical order by the name of the system the SRS is made for. Each system is briefly presented and the results of the questions in the outline for analysis (table 2) of the SRS are shown. Some examples of the different types of requirements are presented as well. The results organized by each question in the table are analyzed and discussed in chapter 5. The web address for each SRS is specified in Appendix A.

4.1 APAF

APAF stands for ASPERA-3 Processing and Archiving Facility. ASPERA-3 is an instrument package for a space mission to Mars and the APAF is a system which is designed to process the telemetry collected by ASPERA-3. The software receives the data from ASPERA-3, processes it, distributes it, presents it on a web-display and finally submits it for storage.

Version of document: 1.0 (2007)

Structure of SRS:	1. Scope 2. Requirements specification descriptions 3. Requirements 4. Notes
Language:	Natural
Categorization of requirements:	1. capability/functional requirements 2. external interface requirements 3. internal interface requirements 4. internal data requirements 5. security & privacy requirements 6. computer resource requirements 7. logistics-related requirements 8. delivery requirements 9. other requirements considered
Identification of requirements:	Letter-number combinations
Number of different categories of requirements:	Functional: 28 Usability: 0 Reliability: 0 Performance: 0 Supportability: 2 Design restrictions: 4

What in IEEE 830 is section two – overview – is here included in the first section. The second section is called “requirements specification description” with different requirements for requirements. Section three is the list of requirements, as in IEEE 830. Following is one section called “notes”.

Examples of functional requirements:

- *APAF-FR-02 The APAF system shall process all ASPERA-3 science data into IDFS data sets.*
- *APAF-FR-07 Web-based displays of the most current ASPERA-3 data shall be provided for public view.*

Example of supportability requirement:

- *APAF-LR-02SwRI shall provide software support for the APAF system*

4.2 CTBTO_WMO_SEA

CTBTO_WMO_SEA (Comprehensive Nuclear Test-Ban Treaty Organization World Meteorological Organization Special Event Analysis) is a software package designed to a) initialize and forward calculations from a larger system and b) provide a web-based interface for collecting, comparing, source locating and reporting data received from meteorological centers.

Version of document: 1.0 (2009)

Structure of SRS:	1. Scope 2. References 3. Functional requirements 4. Acceptance testing requirements 5. Documentation requirements 6. Security requirements 7. Portability requirements 8. Performance requirements 9. Terminology Glossary, abbreviations and appendices
Language:	Natural
Categorization of requirements:	1. functional requirements 2. acceptance testing requirements 3. documentation requirements 4. security requirements 5. portability requirements 6. performance requirements
Identification of requirements:	Hierarchical numbers
Number of different categories of requirements:	Functional: 92 Usability: 6 Reliability: 0 Performance: 5 Supportability: 3 Design restrictions: 2

The SRS is structured to follow an organization-specific template. The introduction, section 1, is quite similar to IEEE 830. Section two is not an overview of the system, but a list of references to related documents. The requirements are not collected under one section, instead different types of requirements have their own headings.

Examples of functional requirements:

- *1.3.2 The software shall allow for post processing of the virtual RN station measurements*
- *2.2.2 The software shall monitor the automated collection of the data received in response to the request.*
- *2.2.4 The web tool shall be capable to generate WMO Centre comparison statistics: Consolidation of auto-reporting on the web page of the exercise, including plot generation, and integration of the standard display of MMFORs inter-comparison statistics published in [9]*

Example of performance requirement:

- *18.4 A full daily update of the reporting web page shall not take longer than 12 hours per 10x10 SRS data request examined.*

Example of design restriction:

- *3.1 The CTBTO_WMO_SEA software shall run under UNIX/LINUX while making use of the native C and FORTRAN compiler package gcc running at the PTS.*

4.3 EVLA CB

This system is an in-between system and the main component of a data processing pipeline in a system called EVLA (the VLA Expansion Project). The CB stands for Correlator Backend. The system is supposed to receive, assemble, format, process and finally deliver data in a suitable way. Data is sent to EVLA CB from a monitor and control system, and after processing it the EVLA CB delivers it to an end-to-end system. This SRS follows IEEE 830, which is also stated on page three in the document.

Version of document: 2.0 (2002)

Structure of SRS:	1. Introduction 2. Overall description 3. Specific requirements
Language:	Natural
Categorization of requirements:	1. external interface requirements 2. performance requirements 3. reliability/availability 4. serviceability 5. maintainability 6. scalability, security 7. installation & upgrades 8. documentation.
Identification of requirements:	Hierarchical numbers
Number of different categories of requirements:	Functional: 66 Usability: 2 Reliability: 8 Performance: 12 Supportability: 13 Design restrictions: 0

Examples of functional requirements:

- *3.2.1.1 Monitor and Control System – The BE shall acknowledge receipt of all data received from M&C*
- *3.2.2.7 Data Invalid – The BE shall replace all invalid data with zero values.*
- *3.2.2.31 Reboot network – The BE shall be able to initiate a reboot of any internal network.*

Example of performance requirement:

- *3.3.2.1 Input – The BE System shall be capable of accepting an aggregate data input stream from the Correlator of a minimum om 1.6 Gbytes/sec. This must be done simultaneously with the output stream, but not necessarily over the same interconnects. This is an initial deployment specification and will be increased over time.*

Example of supportability requirement:

- *3.6.1 Software tools – software tools and pre-built applications that do not have source code available shall come with a complete diagnostic package and customer support*

4.4 I-15 RLCS

The Interstate 15 Reversible Lane Control System (I-15 RLCS) is designed to control the opening and closing of reversible lanes on the Interstate 15. It was developed since the previous system was getting old and was impossible to extend. I-15 RLCS also provides a graphical user interface, process control and monitoring, sequencing, data processing and security as well as reporting.

Version of document: unknown (2004)

Structure of SRS:	IEEE 830: 1. Introduction 2. Overall description 3. Specific requirements
Language:	Natural
Categorization of requirements:	1. external interface requirements 2. functional requirements 3. performance 4. logical database requirements 5. design constraints 6. RLCS application software attributes.
Identification of requirements:	Hierarchical numbers
Number of different categories of requirements:	Functional: 85 Usability: 0 Reliability: 10 Performance: 20 Supportability: 3 Design restrictions: 6

Examples of functional requirements:

- *3.2.2.1 The RCLS shall monitor all field device sensors, and shall process operator requests for changing field device status.*
- *3.2.2.6.1 The RCLS software shall initialize each control unit and device sensor as it is identified.*

Examples of performance requirements:

- *3.3.1.1. The external server data store containing RLCS status for use by external systems shall be updated once per minute*
- *3.3.4.1 At a minimum of every 60 seconds, the system shall check the current date and time against a list of scheduled events for the current mode to determine if any event should be executed.*

Examples of design restrictions:

- *3.5.1.1 The data processing and security, and reporting functions of the RLCS application software shall be implemented with commercial off-the-shelf software*
- *3.5.2.3 The MD5 algorithm shall be used to secure application data and software in the controllers and the application servers.*

4.5 MDOT VII DUAP

MDOT VII DUAP is an abbreviation of Michigan Department of Transportation's Vehicle Infrastructure Integration Data Use Analysis and Processing system. The system is a research program, designed to examine the impact on traffic operations, asset management and transportations planning by new VII data. DUAP is supposed to collect, convert and communicate data to different people to support MDOT.

Version of document: 1.02 (2007)

Structure of SRS:	IEEE 830: 1. Introduction 2. Overall description 3. Specific requirements
Language:	Natural
Categorization of requirements:	1. input services 2. administrative services 3. dynamic data services 4. computational services 5. persistent data services 6. output services 7. presentation services 8. design constraints 9. quality characteristics 10. external services.
Identification of requirements:	Letter-number combinations & level of priority (Low, Medium, High)
Number of different categories of requirements:	Functional: 92 Usability: 0 Reliability: 0 Performance: 0 Supportability: 4 Design restrictions: 6

Examples of functional requirements:

- *IS-010 The System shall collect probe vehicle data. H*
- *IS-070-003 The DUAP data elements shall include roadway event information data fields corresponding to the SAE J2354 structure as enumerated un APPENDIX F – SAE J2354 EventInformation Elements. H*
- *AS-140 The DUAP System shall be able to organize the sequence of execution of computational modes*

Example of supportability requirement:

- *The DUAP System shall be capable of adding new data sources. L*

Examples of design constraints:

- *DC- 010-010 The DUAP System shall use a Java software foundation. M*
- *DC-040 The DUAP System shall use Michigan Geographic Framework geo-references. H*

Most of the requirements in this SRS also have a source (another document) and some of the requirements were commented.

4.6 NPOESS DE

The NPOESS (National Polar-orbiting Operational Environmental Satellite System) Data Exploitation (NDE) is a system which will distribute data from NPOESS observations to civilian customers and operational and climate communities. It is supposed to receive data, process and package it and finally deliver the data to the right people. It is also supposed to be part of customer service.

Version: 1.0 (2007)

Structure of SRS:	1. Scope 2. Referenced documents 3. Requirements 4. Requirements traceability Appendices
Language:	Natural
Categorization of requirements:	1. required states and modes 2. capability requirements 3. external interface requirements 4. internal interface requirements 5. internal data requirements 6. adaption requirements 7. security and privacy requirements 8. computer resource requirements 9. operator-related requirements 10. other requirements
Identification of requirements:	Hierarchical numbers
Number of different categories of requirements:	Functional: 88 Usability: 0 Reliability: 5 Performance: 13 Supportability: 3 Design restrictions: 4

Sections one and three are quite similar to IEEE 830. Section two, which in IEEE 830 is called overview, is sort of included in the first section and instead section two is called referenced documents. After section three which is the list of requirements, there is a fourth section called requirements traceability.

Examples of functional requirements:

- *3.2.1 The System shall be capable of defining Data Products for Ingest*
- *3.3.1 The System shall be capable of receiving data and products from IDPS*

Example of performance requirement:

- *3.4.3 The system shall be capable of executing 99 % of its scheduled tasks in any consecutive 30 day period.*

Example of supportability requirement:

- *3.6.2 The System shall be capable of adding additional capacity without redesign of its infrastructure*

Example of design restriction:

- *3.8.2.2 The System shall be constructed using COTS and Open Source software where it is possible, practical and approved by the Government.*

4.7 OSSAFFCM

OSSAFFCM is an abbreviation of Open Source Sustainability Assessment Framework Format Converter Module. This software is, as the name reveals, a format converter module in an open source framework. The framework is designed to calculate the sustainability for different products and thus see the environmental impact of the product. The purpose of the format converter is to convert data formats from one of four types into another without any loss of data.

Version: 1.1 (2007)

Structure of SRS:	1. Scope 2. Referenced documents 3. Requirements 4. Qualification provisions 5. Requirements traceability 6. Notes
Language:	Natural
Categorization of requirements:	1. required states and modes 2. CSCI capability requirements 3. CSCI external interface requirements 4. CSCI internal interface requirements 5. CSCI internal data requirements 6. adaption requirements 7. safety requirements 8. security and privacy requirements 9. CSCI environment requirements 10. Computer resource requirements 11. software quality factors 12. design and implementation constraints 13. personnel requirements 14. training-related requirements 15. logistics-related requirements 16. other requirements 17. packaging requirements 18. precedence and criticality requirements
Identification of requirements:	Hierarchical numbers
Number of different categories of requirements:	Functional: 19 Usability: 7 Reliability: 1 Performance: 0 Supportability: 5 Design restrictions: 10

What in IEEE 830 is section 2, “overview”, is included in the first section. The second section does instead contain a list of related documents. Section three is the list of requirements, as in IEEE 830. Following are three sections, called “qualification provisions”, “requirements traceability” and “notes”.

Examples of functional requirements:

- *3.2.1.6 The inventory calculation will be able to cope with loops in the product system*
- *3.2.4 The converter will allow conversion between important LCI (Life Cycle Inventory) data formats*

Examples of usability requirements:

- *3.12 The software will be documented in English*
- *3.13.3 The software will be designed to be used by people interested in Life Cycle Assessment and sustainability assessment*

Example of design restriction:

- *3.9 The Framework and the converter will be designed to run in Windows environments (Win 2000, Win XP, Win Vista), Macintosh, and Linux OS. A Java Virtual Machine (JVM) of version 5 or higher needs to be installed. A MySQL database needs to be installed as well. Both JVM and MySQL will be made available on the website*

4.8 SRS2XESAMPLEFLAG

The SRS (Source-Receptor Sensitivity) 2xESampleFlag is an extension to a web-based metrological analysis tool. When a known emitter releases xenon which might affect a xenon sample, the software is designed to flag this sample. SRS2xESampleFlag is supposed to run on any Unix-based system.

Version of document: 1.0 (2009)

Structure of SRS:	1. Scope 2. References 3. Functional requirements 4. Acceptance testing requirements 5. Documentation requirements 6. Security requirements 7. Portability requirements 8. Performance requirements 9. Terminology Glossary, abbreviations and appendices
Language:	Natural
Categorization of requirements:	1. functional requirements 2. acceptance testing requirements 3. documentation requirements 4. security requirements 5. portability requirements 6. performance requirements
Identification of requirements:	Hierarchical numbers
Number of different categories of requirements:	Functional: 49 Usability: 6 Reliability: 0 Performance: 3 Supportability: 4 Design restrictions: 3

The SRS is structured to follow an organization-specific template. (Same as CTBTO_WMO_SEA). The introduction, section 1, is quite similar to IEEE 830. Section two is not an overview of the system, but a list of references. The requirements are not collected under one section, instead different types of requirements have their own headings.

Examples of functional requirements:

- *1.6. The software shall be capable to choose between those available ATM (Atmospheric Transport Modelling) models that were utilized to generate the SRS (Source Receptor Sensitivity) data pertaining to the sample. Further requirements are specified in requirements 5.x*
- *3.1. The software shall be capable to parse the information of the emissions from known xenon sources via a xenon sources inventory input file, parsed by the command line option -e<xenon-emissions-file>.*

Examples of usability requirements:

- *14. All documentation will be written in English in MS Word format*
- *20. There shall be a help function implemented that is called with the* `'-help'` *option and upon any users' choice of options that is non-compliant with the required syntax.*

Example of performance requirement:

- *25. Processing of a 21 days SRS field and preparation of the data shall not last longer than 5 minutes per sample if the full 21 days release (emission) period is examined*

4.9 STEWARDS

STEWARDS is an acronym for Sustaining the Earth's Watersheds – Agricultural Research Data System. It is a system designed to manage data related to water, soil, management and economics. STEWARDS was developed to be part of the Conservation Effects Assessment Project (CEAP) which assesses the environmental effects of a conservation program implementation.

Version of document: 1.1 (2006)

Structure of SRS:	IEEE 830: 1. Introduction 2. Overall description 3. Specific requirements
Language:	Natural
Categorization of requirements:	1. System features 2. External interface requirements 3. Other non-functional requirements
Identification of requirements:	Letter-number combinations
Number of different categories of requirements:	Functional: 85 Usability: 1 Reliability: 1 Performance: 2 Supportability: 2 Design restrictions: 9

Examples of functional requirements:

- *FR-2.3: Browse, query, and download individual sampling station data and metadata Provide access to the data via browsing of sites, stations, and instruments; allow for simple queries to individual datasets; provide a metadata search tool to query dataset parameters; and allow for downloading of datasets (full or partial)*
- *FR-2.5: Generate tabular reports of selected data Provide access to CEAP-related reports, tables and project documents*

Example of performance requirement:

- *PR-2: Loading speed: The data system shall load as quickly as comparable productivity tools on whatever environment it is running on*

Example of supportability requirement:

- *SQ-1: Portability: This database will be built for a particular system and may not be portable but results to queries will be portable between many environments*

Example of design restriction:

- *SR-5: Relational Database Management System – As the primary data storage mechanism for the corporate standard relational database management system, Microsoft SQL Server will be required to support system functionality*

5. Analysis & discussion

This chapter contains a discussion of the above presented results. It is structured to follow the order of the different questions in the outline for analysis presented in chapter two.

5.1 The structure of the different SRSs

Four of the nine SRSs – EVLA CB, I15 RCLS, MDOT VII DUAP and STEWARDS – followed IEEE 830, and in all of these cases, except from MDOT VII DUAP, this was also stated in the SRS. Of the remaining five, four of the SRSs – CTBTO_WMO_SEA, NPOESS DE, OSSAFFCM and SRS2xESampleFlag – were structured in a fairly similar way. These four all started with an introductory section called scope, continued with references or referenced documents in section two, and the requirements began in section three. The table below presents each SRS with an overview of the structure.

Table 5.1. SRSs and their structures.

SRS name	Structure
APAF	1) Scope 2) Requirements Specification Descriptions 3) Requirements 4) Notes
CTBTO_WMO_SEA	1) Scope 2) References 3) Requirements 4) Appendices
EVLA CB	IEEE 830: 1) Introduction 2) Overall Description 3) Requirements
I 15 RLCS	IEEE 830: 1) Introduction 2) Overall Description 3) Requirements
MDOT VII DUAP	IEEE 830: 1) Introduction 2) Overall Description 3) Requirements
NPOESS DE	1) Scope 2) References 3) Requirements 4) Requirements traceability
OSSAFFCM	1) Scope 2) References 3) Requirements 4) Qualification provisions 5) Requirements traceability notes
SRS2XESAMPLEFLAG	1) Scope 2) References 3) Requirements
STEWARDS	IEEE 830: 1) Introduction 2) Overall Description 3) Requirements

In NPOESS DE it was stated that the document is based on the IEEE standard 12207. However, in this standard no given structure for an SRS is described, so if this is true the authors must have used the standard in some other way than following a structure. In this SRS, it was also stated that the requirements management tool DOORS was used to create the SRS. This was the only SRS where it was stated that some kind of tool had been used. No such statement was made in the other SRSs; however, this does not exclude the possibility that some authors might have used such tools.

The last SRS – APAF – was structured almost as the other four in the non-IEEE 830-group, with the exception that section two was called "requirements specification descriptions". The ones who did not follow IEEE 830, including APAF, all had system overview and document overview in the first section. This is quite similar to the sub-headings "system type" and "overview of the subsequent parts of the specification" in IEEE 830. Beyond these two subheadings, all SRSs, both the ones following IEEE 830 and the rest, had other subheadings as well in the first section. These were things like project identification, rationale or definitions.

Another similarity between all SRSs is that they had the specific requirements starting in section three. The CTBTO_WMO_SEA and SRS2xESampleflag turned out to have the same author, which explains why they are structured similarly. The authors for these SRSs follow an organization-specific template, called DSTD, which is stated in the document overview-section of the SRSs. In no other SRSs such information was given.

The biggest differences in the outlines of the SRSs were within the second section and how the requirements were divided into groups. The organization of the requirements is discussed further down. The second section was, as stated above, either "overview", as in IEEE 830, or a section with references to other documents. APAF, which stood out a bit from the others, had a second section that looked nothing like any of the other ones. It was called "requirements specification description", an explanation of necessary attributes for requirements. These attributes were for example that each requirement should have a unique identifier, be necessary for the system or formulated in a clear and concise way. To me, such statements feel a bit superfluous and belong in the requirements engineering literature and not in an SRS. Others seem to agree with me, especially since no other SRSs had such explanations of the nature of requirements.

So, except from APAF, the SRSs appeared to follow either IEEE 830, starting with an introduction, followed by a system overview and then the actual requirements, or another, unnamed structure, starting with scope, followed by a list of referenced documents, and then the actual requirements.

5.2 Language

All SRSs and the requirements in them were written with an informal language. This is consistent with the claim made in chapter two, that contemporary SRSs use natural, informal language. Maybe the use of informal language also is because the systems are not extremely large or complex. Some of them were just small extensions to existing systems, for example the SRS2xESampleFlag or the OSSAFFCM. These "smaller" systems also had a smaller total number of requirements.

5.3 Categorization of requirements

In all cases, the requirements were divided into groups depending on the character of the requirements. These groups were not the same as the ones in chapter two; instead the requirements were grouped differently in each SRS, with the exception of CTBTO_WMO_SEA and SRS2xESampleFlag, who, as mentioned above, have the same

author and thus the same outline for the entire SRS. Except for MDOT VII DUAP, one group of requirements was named almost the same in all SRSs: capability/functional requirements. Another group called external interface requirements was also very common. This group is mentioned by Wiegers (1999) as the first group of requirements in the extended IEEE 830, and appears in six of the nine SRSs. A third group of requirements was also present in six of the nine SRSs, namely security and privacy requirements. Table 5.2 on the next page presents how the requirements were structured in the different SRSs.

In most SRSs, the requirements were divided into 6-10 groups. It would seem logical if systems with lots of requirements had a larger number of groups; however, this is not the case. Interesting to note is that, for example, OSSAFFCM with a fairly small number of requirements had the largest number of groups of requirements (18), while I 15 RLCS, with a great number of requirements, had a much smaller number of groups of requirements (6).

The MDOT VII DUAP stood out from the others since the naming of the groups of requirements was done much differently from the other ones. Most categories were called "services" of some kind, such as input services, administrative services and computational service. Also, most of the requirements in this SRS had a reference document and some of them had comments to help the interpretation of them.

Table 5.2. Categorization of requirements in the different SRSs.

APAF	CTBTO_WMO_	EVLA CB	I 15 RLCS	MDOT VII DUAP	NDE	OSSAFFCM	SRS2XESAMPLE	STEWARDS
1 Capability or Functional requirements **2** External Interface Requirements **3** Internal Interface Requirements **4** Internal Data Requirements **5** Security and Privacy Requirements **6** Computer Resource Requirement **7** Logistics-Related Requirements **8** Delivery Requirements **9** Other Requirements Considered	**1** Functional requirements **2** Acceptance testing requirements **3** Documentation requirements **4** Security requirements **5** Portability requirements **6** Performance requirements	**1** External interface requirements **2** Functional requirements **3** Performance requirements **4** Reliability/ Availability **5** Serviceability **6** Maintainability **7** Scalability **8** Security **9** Installation and upgrades **10** Documentation	**1** External interface requirements **2** Functional requirements **3** Performance **4** Logical database requirements **5** Design constraints **6** RLCS application software attributes	**1** Input services **2** Administrative services **3** Dynamic data services **4** Computational services **5** Persistent data services **6** Output services **7** Presentation services **8** Design constraints **9** Quality characteristics **10** External requirements	**1** Required states & modes **2** Capability requirements **3** External interface requirements **4** Internal interface requirements **5** Internal data requirements **6** Adaption requirements **7** Security and privacy requirements **8** Computer resource requirements **9** Operator-related requirements **10** Other requirements	**1** Required states and modes **2** Capability requirements **3** External interface requirements **4** Internal interface requirements **5** Internal data requirements **6** Adaption requirements **7** Safety requirements **8** Security and privacy requirements **9** Environment requirements **10** Computer resource requirements **11** Software quality factors **12** Design & implementation constraints **13** Personnel requirements **14** Training-related requirements **15** Logistics-related requirements **16** Other requirements **17** Packaging requirements **18** Precedence and criticality requirements	**1** Functional requirements **2** Acceptance testing requirements **3** Documentation requirements **4** Security requirements **5** Portability requirements **6** Performance requirements	**1** System features **2** External interface requirements **3** Other non-functional requirements

When analyzing the documents, it was sometimes quite difficult to identify what type of requirement was stated. The non-functional ones were occasionally very tricky to categorize into Eriksson's (2007) categories. None of the SRSs used the same categorization as Eriksson (2007), which made it necessary for me to categorize the requirements from the category given in the SRS to the, in my opinion, corresponding category in the outline for analysis. However, in most cases the name of the group of requirements corresponded to one of the FURPS+ components or one if its keywords, which made it easier to place them in the right category. For example, EVLA CB had groups of requirements called "serviceability" and "maintainability", which both are FURPS+ keywords for supportability. Functional requirements were sometimes also necessary to identify. The category "security and privacy requirements" which appeared in six of the nine SRSs were usually stated under a heading of its own. Since "security" is one of the FURPS+ keywords for functionality and the requirements usually followed a noun-verb construction such as "*the System shall be capable of generating backups for all NDE data, procedures, and software"* (requirement 3.7.2 in NPOESS DE), they were counted as functional requirements. However, in STEWARDS, the category "security requirements" is listed as a subcategory of "other non-functional requirements". When looking at the specific requirements, there was for example one that looked like this:

> *SCR-4: Availability*
> *The fourth consideration for security requirements is availability. The system must be available to the intended audience 24 hours per day, 7 days a week with, 99% availability and a tolerance of -5% (not less than 50% of working hours in any week). For this system, availability will be concerned with the reliability of the software and network components. Intentional "denial of service attacks" is not foreseen as a significant concern.* (STEWARDS p. 13)

This one is obviously a non-functional requirement, which in my categorization belongs to the group "reliability". This example shows that the categorization of requirements can differ a lot.

To sum up this paragraph, no SRS had the same structure of the requirements as the one in the theoretical background in chapter two. Instead, all SRSs had individual organizations of the requirements, except from CTBTO_WMO_SEA and SRS2xESampleFlag who had the same organization of requirements, probably because they have the same author. Because of the great variety of ways to organize the requirements into groups, I had to personally re-categorize the requirements to fit them in the outline for analysis and be able to compare results. This had to be a sometime rather interpretative activity, and if someone else was to do it the results might be slightly different, especially in the counting of the requirements.

5.4 Does each requirement have its unique identifier?

All requirements had unique identifiers. In most cases the requirements were hierarchically numbered, but in APAF, MDOT VII DUAP and STEWARDS there were letter-number combinations. This is consistent with the statement both Wiegers (1999) and Eriksson (2007) makes; that hierarchical numbering is the conventional way to label requirements. No other types of labeling were present in the SRSs analyzed in this thesis.

5.5 What is the most frequent category of requirements?

Functional requirements were much more frequent than non-functional ones and design requirements. This confirms the statement made earlier. It was also stated that the reason for this is that non-functional requirements are difficult to evaluate and measure, and because they have many aspects to consider. Another claim made earlier was that functional requirements are specified clearer. In some cases, the functional requirements were extremely detailed, while there were only a few non-functional requirements in the same document. This was particularly noticeable in CTBTO_WMO_SEA, where the functional requirements sometimes were very detailed with a great number of sub-requirements: *1.2.1.4.1.1.1. For the nuclide it shall be possible to enter the nuclide symbol name explicitly (e.g. Xe-133). The name will be translated into the nuclide specific half-life time (via a lookup table offered by the commission). If 'Tracer' is given, a zero half-life time shall be applied (indicated by a '999.9' in the configuration file 'SPECIES')* while a non-functional requirement or design restriction could be as short as this: *12 Source code will be documented in the code according to suitable standards.* However, these two requirements are, especially the latter one, rather extreme and should not be viewed as facts, only as examples or end-points of a scale on how detailed requirements can be. Most of the time, functional and non-functional requirements seemed to have been given the same level of detail.

Eriksson (2007) divided non-functional requirements into four groups; usability, reliability, performance and supportability. Of these four groups, performance requirements were the most numerous ones. In four of the SRSs, performance requirements were labeled as an own group of requirements. This was the only group of non-functional requirements that was consistent with Eriksson's classification. As discussed above, the division of requirements into different groups in each of the SRSs seemed to follow individual templates.

Design restrictions were not very numerous. Perhaps they are not formulated to a high extent in the SRSs because such details are described in another document. For example, in STEWARDS there is a reference to a site called "USDA Web Style Guide", MDOT VII DUAP has a reference to a document called "Systems Engineering Methodology version 1.0" and in the SRS for EVLA CB there is a reference to a document called "EVLA Correlator Monitor and Control System, Test Software and Backend Software Requirements and Design Concepts".

Regarding the non-functional requirements, performance requirements were the most frequent ones. Maybe this is because of the types of non-functional requirements, performance requirements are the easiest to measure, since they are often expressed with time. For example, take the requirement 3.10.2.3 from NPOESS DE: *The System shall deliver A-DCS telemetry data from IDPS to the US Global Processing Center within 1minute of their receipt.* Now compare this to the requirement 3.11.2 from OSSAFFCM: */3.11.2/ The software will be easily learned and used. This usability will be supported by documentation accompanying the software.* How easy is that to measure? As Avison & Fitzgerald (2006) states, nonfunctional requirements can be hard to evaluate, especially ones that are not about time.

I have not discussed prioritizing requirements in this thesis. However, an interesting thing to note that concerns both priority and importance of requirements is that all MDOT VII DUAP software quality characteristics were marked "L" – not important or low priority. This confirms that non-functional requirements are not considered as important as functional ones.

Wrapping up, functional requirements were much more common than non-functional ones and design restrictions. This confirms the statement made in the introduction and again in chapter two, that functional requirements are the most frequent ones in SRSs. Sometimes the functional requirements were more detailed than non-functional ones, but this was only apparent in special cases and not something that happened consistently. According to the results in this thesis, it is not true that functional requirements are described clearer than non-functional ones or design restrictions. There are, as shown above, examples that confirm this, but these are exceptions, not the rule.

6. Conclusions

The purpose of this thesis is to identify similarities and differences in SRS composition and to illustrate what type of requirements are most frequent. In order to do this, this thesis started out with raising two main research questions. The first question considered the structure of SRSs and the organization of the specific requirements, and the second one was to find out which one the most common type of requirements is.

From the SRSs collected for the analysis in this thesis, there seems to be two main types of structures for SRSs. They either follow IEEE 830 with its introduction – overview – list of requirements structure, or have an outline which is introduction – references – list of requirements. There was no similar pattern regarding the organization of the actual requirements. With the exception of CTBTO_WMO_SEA and SRS2xeSampleFlag who have the same origin and were structured the exact same way, no other two SRSs had a similar way to organize the requirements. The only similarity about the way the requirements were organized was that in all cases, the requirements were divided into groups, and that they had unique identifiers. Three groups of requirements were frequently used: functional/capability requirements, external interface requirements and security/privacy requirements. The differences in SRS composition are, however, not very extreme. This is not consistent with the results from Power's (2002) thesis mentioned earlier. Perhaps another collection of documents would give different results, but from these findings, SRSs are quite alike.

Looking at the specific requirements, functional requirements outnumber the other categories by large. Even combined, there were a lot less non-functional requirements than functional ones. This does confirm the claim made in chapter two is probably not completely fair. It might have been better to follow Wiktorin (2003) who stated that it can be good to divide functional requirements into groups. There were quite a few requirements in the SRSs which I regarded as functional requirements, even if they were not in the specific functional/capability requirements group in the SRS. Since the functional requirements category thus became much broader than the different categories of non-functional requirements, it was not very surprising that there were many more functional requirements.

Something that could not be given an answer to by analyzing the SRSs was reasons for the choice of structure or organization of requirements. For example, the OSSAFFCM which had the largest number of groups of requirements had a very small number of actual requirements. Personally, I find it unnecessary to have so many groups of requirements when in some cases there was only one or two requirements in each category, but maybe there is a good reason for it. A possible suggestion for further research would then be to pick out a couple of SRSs and contact the originators to get the answer on questions like this.

Following standards, like the IEEE 830, probably simplifies the documentation of requirements. If every requirements engineer needed to document requirements in his/her own way, the development team would have to put a lot of time in interpreting the SRS for every new project. This is of course why standards are developed in the first place. Personally, I believe standards are useful, as long as they are applicable to the organizational context and allow modifications. Another benefit of using standards or templates is that they do not need to be organization-specific. If the development team works independently and is not bound to one specific company, they can use the same template for different projects in different organizations.

Regarding the specific groups of requirements, it might be possible to use templates as well. The literature showed a number of different ways to categorize requirements (e.g. Avison & Fitzgerald, 2006; Eriksson, 2007; IEEE 1998a; Wiktorin, 2003), and the results in this thesis showed a great variety in the organization of specific requirements. A more general classification might be useful, particularly for independent development teams, or when development teams from different companies cooperate in projects. However, these are mere speculations, but as stated above, standards can be useful and are apparently being used, at least to some extent.

Appendix A: Abbreviations

Abbreviations of SRS names:

APAF: ASPERA-3 Processing and Archiving Facility

CTBTO_WMO_SEA: Comprehensive Nuclear Test-Ban Treaty Organization World Meteorological Organization Special Event Analysis

EVLA CB: the VLA Expansion Project Correlator Backend

I-15 RLCS: Interstate 15 Reversible Lane Control System

MDOT VII DUAP: Michigan Department of Transportation's Vehicle Infrastructure Integration Data Use Analysis and Processing system

NPOESS DE: National Polar-orbiting Operational Environmental Satellite System Data Exploitation

OSSAFFCM: Open Source Sustainability Assessment Framework Format Converter Module

SRS2xESampleFlag: Source-Receptor Sensitivity 2xE Sample Flag

STEWARDS: Sustaining the Earth's Watersheds – Agricultural Research Data System

Other abbreviations:

FURPS+ : Functionality, Usability, Reliability, Performance, Supportability

IEEE: Institute of Electrical and Electronics Engineers

ISO: International Organization for Standardization

SRD: Systems requirement determination

SRS: Software/systems requirement specification

VDM-SL: Vienna Development Method Specification Language

Appendix B: SRS Sources

All websites visited January 3, 2010

ASPERA-3 Processing and Archiving Facility
http://www.aspera-3.org/idfs/APAF_SRS_V1.0.pdf

CTBTO_WMO_SEA
http://www.ctbto.org/fileadmin/user_upload/procurement/2008/RFP2009-0339-CTBTO_WMO_SEA_Software_Requirements_Specification-DISCHENDORF.pdf

EVLA Correlator Backend
http://www.aoc.nrao.edu/evla/techdocs/computer/workdocs/Corr_bkend_Req_Soft.pdf

I 15 Reversible Lane Control System
http://www.dot.ca.gov/dist11/operations/I15RLCS/RFP_DOT2040_SecVID_Apr21_2004.pdf

MDOT Vehicle Infrastructure Integration Data Use Analysis and Processing (VII DUAP)
http://www.michigan.gov/documents/mdot/MDOT_DUAP_SysReq_Final_220099_7.pdf
Permission to use document given via e-mail by Colleen LoVette, Michigan Department of Transportation (lovettec@michigan.gov) on Nov 19, 2009

NPOESS Data Exploitation
http://projects.osd.noaa.gov/NDE/pub-docs/SystemRequirementsDoc.pdf

Open Source Sustainability Assessment Framework, Format Converter Module
http://www.openlca.org/uploads/media/Software_Requirements_v1.1_3Feb07.pdf

SRS2xESampleFlag:
http://www.ctbto.org/fileadmin/user_upload/procurement/2008/RFP2009-0337-SRS2XeSampleFlag_Software_Requirements_Specification-DISCHENDORF.pdf

STEWARDS
ftp://ftp-fc.sc.egov.usda.gov/NHQ/nri/ceap/stewardssystemdesign030206.pdf

Reference list

Avison, D. & Fitzgerald, G., 2006. *Information Systems Development: Methodologies, Techniques & Tools.* UK: McGraw-Hill.

Cysneiros, L. M. & Leite, J. C. S. P., 2004. Nonfunctional Requirements: From Elicitation to Conceptual Models. *IEEE Transactions on Software Engineering,* 30(5), pp. 328-350.

Dahlstrand, M., Fredborg, H. & Leandersson, S., 2009. *Co-design av co-design – förslag på riktlinjer för arbetssättet och utformningen av kravspecifikationer.* [online] Bachelor thesis. Borås: University of Borås. Available at: http://bada.hb.se/bitstream/2320/5193/1/2009KI12.pdf [Accessed Nov 25, 2009]

Daniels, J., & Bahill, T., 2004. Requirements and Use Cases. *Systems Engineering,* 7(4), pp. 303-319.

Drolsch, G., 2000. *Formal Specification of a Security System Module in VDM-SL.* Technicn

Duggan, E.W. & Thachenkary, C.S., 2003. Higher Quality Requirements: Supporting Joint Application Development with the Nominal Group Technique. *Information Technology and Management*, 4(4), pp. 391-408.

Eriksson, U., 2007. *Kravhantering för IT-system.* Malmö: Studentlitteratur.

Franko, S. & Hansson, M., 2006. *Verksamhetsregler vid utformning av kravspecifikation.* [online] Bachelor thesis. Lund: Lund University/School of Economics and Management. Available at: http://biblioteket.ehl.lu.se/olle/papers/0002030.pdf [Accessed Nov 28, 2009]

Grady, R.B., 1992. *Practical software metrics for project management and process improvement.* Englewoods Cliffs, N.J: Prentice Hall.

Halvorsen, K., 1992. *Samhällsvetenskaplig metod.* Lund: Studentlitteratur.

Ho, C., Williams, L. & Antón, A. I., 2007. Improving Performance Requirements Specifications from Field Failure Reports. *15th IEEE International Requirements Engineering Conference,* pp. 79-88.

Hull, E., Jackson, K. & Dick, J., 2005. *Requirements Engineering.*[e-book] Available at http://dx.doi.org/10.1007/b138335 [Accessed Dec 29, 2009]

IEEE, 1998a. *IEEE STD 830-1998: IEEE Recommended Practice for Software Requirements Specifications.* USA: The Institute of Electrical and Electronics Engineers.

IEEE, 1998b. *(ISO/IEC 12207) Standard for Information Technology – Software Life Cycle Processes.* USA: The Institute of Electrical and Electronics Engineers.

Machado, R. J., Ramos, I. & Fernandes, J. M., 2005. Specification of Requirements Models. In Aurum, A. & Wohlin, C, ed. *Engineering and Managing Software Requirements.* [e-book] Available at http://www.springerlink.com.ludwig.lub.lu.se/content/t3v7w4/ [Accessed Dec 27, 2009]

Maxwell, J. A., 2005. *Qualitative Research Design: An Interactive Approach.* Thousand Oaks: Sage Publications, Inc.

Nicolás, J. & Toval, A., 2009. On the Generation of Requirements Specifications from Software Engineering Models: a Systematic Literature Review. *Information and Software Technology,* 51, pp. 1291-1307.

Power, N. M., 2002. *A Grounded Theory of Requirements Documentation in the Practice of Software Development.* Ph D thesis. Dublin City University: School of Computer Applications.

Smith, S., Lai, L. & Ridha, K., 2007. Requirements Analysis for Engineering Computation: A Systematic Approach for Improving Reliability. *Reliable Computing,* 13, pp. 83-107.

Svenning, C., 2003. *Metodboken,* 5th ed. Eslöv: Lorentz förlag.

Wiegers, K. E., 1999. *Software Requirements.* Washington: Microsoft Press.

Wieringa, R. J., 1996. *Requirements Engineering: Frameworks for Understanding.* Chichester: John Wiley & Sons Ltd.

Wiktorin, L., 2003. *Systemutveckling på 2000-talet*. Lund: Studentlitteratur.