Writing Requirements

JAN BARKHED

Writing clear and understandable software requirements

J|B

Writing Requirements

Patterns and rules for writing clear and understandable requirements

Foreword

When I gave my first training class in Requirements Engineering back in 1996, I had a very different approach to how engineers ought to learn the subject. I believed it was important to understand Requirements Engineering from a process oriented perspective. How else, I thought, can engineers put techniques and methods in a context that made sense? My pupils soon put me on other thoughts. They were only interested in techniques, simple techniques that gave proven results. They were not interested in knowing that a technical requirement had seven quality attributes, or that a development team could use open ended interviews to conduct Requirements elicitation. They wanted to know exactly how to write clear requirements that were testable, and it should be easy to learn, and they didn't have much time to learn it.

I wrote this guide, for any one to use as they saw fit. It is a kind of cookbook. It can to advantage be used when conducting reviews of Requirements Specifications. A project or development team can use this guide to write their own checklists. Even project managers can use patterns and rules from this guide when they are writing their Project Specification. After all, project goals need to be clear and understandable as well.

Most examples come from real projects that produced real products and solutions for real customers. The examples have, however, been modified to hide their origin.

The first version of this book was written in 2002, but it wasn't released as a downloadable e-book until 2006. Every rule found in this book is based on advice given in technical writing, and on generally accepted guidelines for how to write good literature.

This guide is not about how to find requirements for a specific technical domain. It does not describe on what level engineers should write requirements, or how they should structure a Requirements Specification, although this is important also. It is solely about style and form. The content is up to the writers to find.

Chapter 1

Natural Language

WRITING REQUIREMENTS IN NATURAL LANGUAGE

The purpose of this publication is to give readers and practitioners some patterns and rules how to write clear and understandable requirements. Requirements Specifications are mostly written in natural language (the language we speak). This is an asset, but it also creates problems, since our language is filled with subtle distinctions and ambiguities, which create misunderstandings and misinterpretations. People can handle this in day to day conversations by using their body language and voice to add information to their words. Misunderstandings due to a written document is not that obvious and harder to handle. A Requirements Specification needs to be clear, and it can not afford to be misunderstood. Unfortunately, authors like to vary their writing; it keeps the reader from falling asleep. Authors like to be unique, exciting and creative, to find elegant formulations. Sadly, most of these qualities suit requirements badly.

DISPOSITION

Each chapter contains a number of examples, how to write and how to not write requirements. Do not feel overwhelmed by all guidelines. They are there to help, not to rule the work. Remember that there are always exceptions to the rule. Patterns are never complete—even rules have exceptions sometimes. Writers and practitioners should therefore use their common sense. All examples are from the technical domain, and from software development. Do not feel discourage by this, if the work is non-technical in its very nature. The examples are chosen because they are easy to understand.

EXAMPLES

Most examples show how to really write requirements. But some examples are given to illustrate how requirements should not be written. The dotted box below illustrates the later.

Do not write like this!

CHAPTER 2

IDENTITY

TAGGING REQUIREMENTS

Each requirement is individually identified by a tag in the Requirements Specification. The tag WA-32-C gives a unique identity to the requirement below.

WA-32-C	The Web application writes error state information to an error text file.

Separation of requirements makes it easier for the members of the project and for the different stake-holders on several levels.

- It improves the readability of the Requirements Specification;
- It improves all out traceability between requirements, sources, design and test;
- It vastly improves the capability to write test cases and conduct testing on all levels.

But separating requirements into individual statements can also have a drawback that impacts understanding. The reader may get a fragmentary or incomplete picture of the behavior and properties of the system. It is therefore important that all requirements in a chapter connects (they should form a coherency). Writing a rationale in the beginning of a chapter helps the reader to understand the context for the requirements that follow. A rationale can also help the reader to understand individual requirements (see the rationales chapter).

THE STRUCTURE OF TAGS

Tags are structured in the following way:

<Domain area>—<serial number>—<revision state>

The <Domain_area> is a letter (or several letters) describing which problem domain the requirement concerns. The metrics generator domain may for example be represented by MG, the database manager domain by DM, and the application manager domain by AM. If all domains are addressed, the letter G for General can by used. The letter R (for requirement) can be used if the domain area identity is not used. There is a risk to use domain area identities. Requirements that are identified to belong to one specific domain are not seldom also valid for other domains. For example can a metrics requirement easily be applicable for the graphics generator and vice versa.

The serial number gives all requirements within a domain area a unique identity. The serial number starts at 1 (one) and is stepped upwards for each new requirement. The same serial number can be used if the requirements are separated by the domain area identity (for example AM-123-A and MG-123-A).

Revisions of a requirement are a change in the content of the requirement. The revision is marked by a new Rev-state. Rev-state is denoted by capital letters stepped in alphabetical order A—Z, AA—ZZ, The letters I, O, Q, R and W must, however, not be used. The first version of a requirement is denoted by A.

TAG USAGE

All requirements for a particular customer or site need to be unique for a project. If there are several sub-projects in the same delivery project, they all need to tag their requirements uniquely, or there may be several G-1-A requirements.

Tags are never reused within a project. If a requirement has been identified once, written into the document and then removed, the tag belongs to the requirement. If the requirement should re-emerge at a later stage, the same tag is used (possibly with a new revision).

CHAPTER 3

FORM

USE ONE OF TWO FORMS

Requirements can be written on two forms. The first form is used frequently in Requirements Specifications, and uses the auxiliary verb "shall":

AM-19-C	The Application Manager *shall* write error messages to an error log-file.

The shall-form generates clumsy constructs like "shall be able to" frequently, which adds no real information to the requirement, although grammatically necessary. This type of construct may falsely give the impression that there is a condition associated with the requirement:

AM-35-B	The Application Manager *shall be able to* handle constraint errors, if an exception occurs.

The if-condition doesn't make sense, and the selected shall-form feels strained. Sometimes the condition is really missing, but more often than not it is never needed. The shall- form may also indicate an intention or purpose: the system shall have the intended behavior in the future. See further the will-form.

The second form is not as common in Requirements Specifications, and it uses the present tense.

AM-19-D The Application Manager *writes* error messages to an error log-file.

The present tense-form is more straightforward and simpler to use. Present tense describes what happens now, in real-time. All actions (verbs) are written in the present tense, for example:

- "The system sends”
- "The system distributes”
- "The issue is displayed”
- "The system has” etc.

The present tense will be used in most examples in this document. The word "shall" is an auxiliary verb, which is almost never used in modern English. The word is more frequently used in conservative English, and was used to draft contracts and legal statements in the old ages.

Americans almost never uses the word "shall" in their language; they use words like "will", "should", "would", "might" etc. Until the ‘90s, the US Department of Defence requested that all Requirements Specifications must be written on the shall-form. This prerequisite has, however, been dropped, and contractors are now allowed to use other forms.

To indicate optionality when it comes to the content of a requirement, the word “can” can be used. The condition of the optionality is unspecified, but it indicates mostly human will or circumstances that lie outside the responsibility of the Requirements Specification.

USING OTHER FORMS

Different forms are often thoughtlessly mixed in the same Requirements Specification. For example:

AM-56-A The Database Client *will* use SSL to communicate with the database.

The will-form may demonstrate that the system is not yet built, and it will therefore have the specified behavior only when it exists. But this is only true during the initial

part of the development work. When the testers execute their test cases and read the Requirements Specification, the system really does exist.

AM-56-A	The Database Client *should* be able to use SSL to communicate with the database.

The should-form may give the impression that the specified behavior should be performed, but not for certain. The should-form is weak and only creates uncertainty with the reader.

AM-56-A	The Database Client must use SSL to communicate with the database.

The must-form is in all aspects identical to the shall-form. The form is more modern than the shall-form, but the form fulfills no other purpose.

Some writers mix these forms on purpose. By doing so, they indicate that certain requirements are optional. As the Requirements Specification is the formal agreement between the contractor (who builds the system) and the customer what is going to be delivered, these types of uncertainties only creates confusion regarding what has been promised and should therefore be avoided altogether. Instead, requirements can be assigned a priority to indicate their importance in the eyes of the customer, for example Non critical, Critical or Very critical.

On the whole, all these forms should be avoided. Use the shall-form or the present tense-form consistently throughout the whole document and keep the writing simple.

Chapter 4

Active or Passive Voice

WRITE REQUIREMENTS IN THE ACTIVE VOICE

Sentences can be written in the active or passive voice. An example of the passive voice is:

> IP-addresses *are leased* to clients in the network.

The same statement using the active voice:

> The DHCP server leases IP-addresses to clients in the network.

Requirements are written in the active voice for the following reasons:

- The passive voice draws attention to the receiver of the action, the Object (in this case the clients in the network). The performer of the action (the Actor - the DHCP server) and the action itself (leases) is generally more interesting than the Object.
- By using the active voice, the performer of the action is stated explicitly in the requirement. Leaving out the Actor may cause serious questions to be unanswered (who is doing this?).
- The passive voice is considered wordy and weak.
- Sentences in the active voice are generally clearer and more direct than those in passive voice.

The passive voice is recognized by sentences with the words:

- is
- am
- are
- were
- was
- been
- being
- be

... and words that ends in:

- -en
- -ed

For some reason, the passive voice is very appealing to writers of requirements. The passive voice is frequently used throughout the Requirements Specification and is generally mixed with the active voice without thought. Of all patterns and rules given by this publication, writers tend to violate the active voice the most.

CHAPTER 5

RATIONALES

GIVING REQUIREMENTS A CONTEXT

Separation of requirements into individual statements may cause the reader to get a fragmentary and incomplete picture of the behavior and properties of the system. Writing a rationale may help to avoid this.

The rationale is written as an introduction to a single requirement or a group of requirements. It serves as a background, and makes the requirements easier to understand. A rationale should be written at least for each chapter, but it can be written more frequently. For example:

Rationale

The user opens the preferences panel in the application menu to set the database connection preferences. All preferences are stored as user defaults, so the user doesn't have to set them again each time the panel is opened. The user password is stored in the Keychain application. In the preferences panel, the user can also search for all active databases on the given IP or computer address.

The rationale may contain:

- Free text
- Pictures
- Tables
- Lists and bullets
- Definitions which are used by the requirements in the chapter. For example:

Rationale

In this chapter, computer address is defined as either the IP address (10.0.0.1) or the fully qualified computer name (gandalf.company.com).

Definitions which are used throughout the whole document are usually written in the Definitions chapter. If there is a need for a local definition, this can be done in the rationale.

The rationale should not contain any vital information that is not part of any requirement. The rational is never tested, and it never serves as an agreement between the development team and the customer. This line can be very fine to walk, and the writer must be watchful not to include anything that can be understood as a requirement (promise) by the customer.

Some authors write one rationale for each requirement. One rationale per requirement can, however, lead to unnecessary repetition of information. When there is a real need for an explicit rationale. the rationale can be written before the requirement. For example:

DM-271-B The user can select to use SSL to connect to the database.

Rationale

When the user stores the XML-specification in the database, the Database Manager makes a number of checks to make sure that the document is not already stored and that all information that needs to be given is present. If any of these checks fail, the application informs the user of the failure or failures. The Database Manager does not store the document in case of failure.

DM-94-G When the user stores an XML-document, the Database Manager asserts that the document number and revision does not exist in the database.

Chapter 6

Conjunctions

ONE REQUIREMENT MAY BE SEVERAL REQUIREMENTS

Conjunctions joins sentences, clauses, phrases, or words. The most common conjunctions when writing requirements are:

- and
- or

For example:

LC-3-C	The Operator can enable *or* disable the calculation of a TCT value. The Operator can choose an ANDON station *or* an assembly line section.
B-46-C	A user of the application belongs to one *or* several of the following authority groups: • Operator; • Maintenance Engineer; • Manager.

There are more subtle conjunctions, for example

- but
- nor
- for
- so
- either ... or
- neither ... nor
- both ... and
- not only ... but
- partly ... partly
- yet

Conjunctions may suggest that a requirement is actually several requirements. For example:

MT-197-A The Metrics Generator stores general *and* specific measurements.

The MT-197-A requirement may easily be written as two requirements. A more obvious example is:

M-52-A The Metrics Generator displays N/A for not applicable measurements, *and* colors out of bound measurements in red.

Co-ordination of actors and objects with "and" (more seldom with "or") is more acceptable than co-ordinating actions. Co-ordinating actions may result in long sentences that are hard to read. In reality, requirements can live without most conjunctions; the conjunctions "and" and "or" are acceptable in some cases. Some conjunctions are also necessary to express time dependent (temporal) conditions (see subordinate clauses).

Chapter 7

Parenthesis and Special Characters

USE PARENTHESIS WITH CARE

Some authors like to use parenthesis, they fertilize with them. But parenthesis should be used with care. They are as risky to use as subordinate-clauses, because they divert the reader's attention from the core meaning of the requirement. Parentheses are sometimes used to explain once more what the author already has written. If parentheses are used in this way, the author has failed to put into words the intended meaning and should rewrite the requirement. The following requirement is an example of how to state the same subject twice, without making it much clearer the second time.

> TS-40-A The Ticket system adds an additional charge (a fixed amount to be added per travel zone, which is time independent) per travel zone.

It is also common to put examples within parenthesis, such as requirement TS-329-A. But examples are never complete (or they wouldn't be examples), which means that the requirement is open for interpretation. See the examples chapter on how to use examples.

> TS-329-A The Ticket system uses the "Day and night" tariff class for all exception dates (e.g. 2005-11-01, 2005-11-11 etc.).

Instead, parenthesis should be used in requirements to:

- Explain abbreviations
- Define an abbreviation
- Explain foreign words

AS-130-D	The Accounting system uses Polish New Zlotych (PLN) as base currency.

The following requirement uses the parenthesis to explain what the English word Recipient is in Polish.

AS-353-B	The invoice page contains a Recipient (Adres Odbiorcy) section.

THERE IS NO NEED FOR SEMICOLON

Authors commonly use the semicolon (;) in the wrong way, probably because they think the semicolon is in between colon (:) and comma (,). This is wrong. An incorrect usage of semicolon is shown in requirement AS-117-F:

AS-117-F	The Accounting system has five VAT rates; 22%, 7%, 0%. VAT exempt, and Not VATable.

Semicolon can be used in two ways:

- It separates items in a list after a colon;
- It separates two independent clauses in a sentence.

For example:

AS-133-E	The Accounting system marks the invoices with the words: FAKTURA VAT ORYGINAL for the original invoice; KOPIA for the invoice copy; DUPLIKAT ORYGINAL for the duplicate original invoice; and DUPLIKAT KOPIA for the duplicate copy invoice.

Requirement AS-133-E would probably be better formulated with a bulleted list, but the example is given to illustrate how semicolon should be used.

The author can use the semicolon if two independent clauses belong together. The semicolon is not as hard as a period (.), but it is more distinct than a comma (,). For example:

AS-2-D	The Accounting system contains one object type; the object type is Marketing.

Make the following test: if the semicolon can be replaced with a period and a new sentence, the semicolon is correctly used. In requirement AS-2-D the two clauses can be considered as close, because both are about the object type.

In reality, there is no need to use semicolons when writing requirements. This chapter only illustrates how very frequently authors misuse the character.

FIGURE 7.1 Programming languages and SQL languages are a subset of natural language.

Chapter 8

Examples

EXPLAINING REQUIREMENTS THROUGH EXAMPLES

The author may sometimes feel that a requirement is not clear enough, no matter how many times it has been rewritten. Writing an example may help in these cases.

AS-43-D	The Accounting system operator types the wild-card character '?' in the account field of the verification window to search for an account number.

Example:
A query of "40?" in the account field of the verification window will return a list with the following hits: 4000 Retail Sales; 4001 Wholesale Sales; 4002 Sales-Service.

Some writers would say that using an example to explain what was just stated in the requirement is an obvious failure, and actually invalidates the requirement itself. If, however, examples are used with discrimination and not as an excuse for badly written requirements, this concern may be uncalled for.

Examples consist of free text, pictures, tables and lists. The example is strictly not part of the requirement itself. It is additional information to explain the requirement, the same way a rationale explains a chapter. We can think of the requirement as a mathematical proof: it is general but precise to its nature, and it has clear boundaries. To explain the proof to the reader, the writer sometimes exemplifies with a case. But the example is never part of a proof.

This means the example text may be changed without changing the revision of the requirement.

Chapter 9

References

REFERENCES CAN MAKE REQUIREMENTS INVALID OR UNCLEAR

References are used in a Requirements Specification to tell a reader that more information is available on a specific subject. But references can also create confusion and erode the structure of a document, especially when references are not updated. An invalid reference is a source of error. In general, references create dependencies that are hard to survey, and lots of references may indicate that there is something wrong with the document structure.

The following rules apply to the usage of references in a Requirements Specification:

1. *Do not use references between requirements*. The context of a requirement is given by where it is written in the Requirements Specification, i.e. the chapter heading, paragraph text, and chapter rationale. A referenced requirement that is removed or rewritten may cause a chain reaction of revisions to other requirements. A reference to a removed requirement may in worst case become a "dangling reference".

2. *Reference to tables, figures and appendixes from within a requirement is acceptable*. The tables, figures and appendixes should all be within the document.

3. *Do not reference from within a requirement to a chapter*. Modern word processing programs update chapter numbers without the knowledge of the author. Requirements are therefor changed without the revision letter being stepped. If a reference to another chapter is needed, this can be written in the rationale.

4. *Never reference from within a requirement to other documents*. The Requirements Specification is often under full configuration control within a project. Once baselined, a Requirements Specification can only be changed through formal change handling. Sometimes writers feel it is more convenient to put information into a document that is referenced from the Requirements Specification, but not under configuration control. This way, changes can be made without too much administrative work. This way of working only fools the system, although the writer may not understand it at the time. It only serves to sabotage the configuration control of the project. There is only one exception to this rule; if the other document is also a Requirements Specification. It is still all right to reference any document from within the Requirements Specification (except from within the requirements themselves), provided the information in the referenced document is not treated as requirements.

All these references below are acceptable to use:

TS-329-A	The adult ticket fare uses the "Day and night" tariff class for all exception dates in *Appendix M*.

Reference 12 below is another Requirements Specification. The format in this specification is crucial for the AS-253-A requirement to be fulfilled.

AS-253-A	The accounting system produces a VAT report with a format according to ref. [12].

The following requirement uses a reference to a table, which is attached to the requirement.

TS-328-A The adult ticket fare uses a Tariff Zone Plan with a content according to Table 1.

Table 1.

ZONE	TARIFF
Red	2
Green	3
Yellow	4
Orange	5
Blue	6
White	7
Black	8

Chapter 10

Bullets and lists

BULLETS AND LISTS GIVE STRUCTURE

Readers like bullets and lists, because they give structure to the reading and are easy to understand. For these qualities, it is recommended to use bullets and lists within requirements.

But bullets and lists can be viewed as a form of conjunction (see the chapter about conjunctions), and they should therefore be used with care. Below is an example of an acceptable requirement with bullets.

AS-17-E The accounting system has five VAT rates:

- 22% VAT
- 7% VAT
- 0% VAT
- VAT exempt
- Not VATable

The following requirement is also acceptable.

SQ-6-B A sequence-rule can define:

1. The batch size;
2. A starting mix number;
3. Every Nth car;
4. An ending mix number.

Requirement AS-49-C is however not acceptable, as it is an example of requirements amalgamation (several requirements are merged into one requirement). It is easy to split AS-49-C into 4 or 5 separate requirements.

AS-49-C The accounting system produces invoices with the following properties:

- The invoice has an invoice number.
- The invoice currency is USD.
- The invoice language is English.
- The invoice contains the VAT amount and the total sum to pay.

Chapter 11

Tables

TABLES GIVE STRUCTURE

Tables can easily be used to give structured information to the reader.

MT-4-B The Metrics Generator stores the metrics specified in Table 2:

Table 2.

METRIC	UNIT	CHART TYPE
Number of product requirements	requirement	MAMR-chart
Number of traceable product requirements	requirement	Attributes data
Number of source lines of code	source line of code	Variables data
Number of document pages	page	Variables data
Number of defects discovered	defect	Attributes data

Large tables can be specified in the Appendix and be referenced from the requirement.

GG-42-A The Graphics Generator draws metric chart types as specified in Appendix E.

Chapter 12

Appendixes

APPENDIXES CAN BE USED TO STORE BULKY INFORMATION

Appendixes can be used to store large tables, figures, layouts of reports and printouts, and other bulky information. The table/figure/layout must be referenced from within the requirement. The Appendix is in every respect a requirement, but it must be referenced from within one or several tagged requirements. Appendixes, which contain requirement information, can not exist without a referencing requirement. Appendixes that are not referenced from within a requirement are not considered as requirements.

Figure 12.1 Authors can write bulky information in an Appendix.

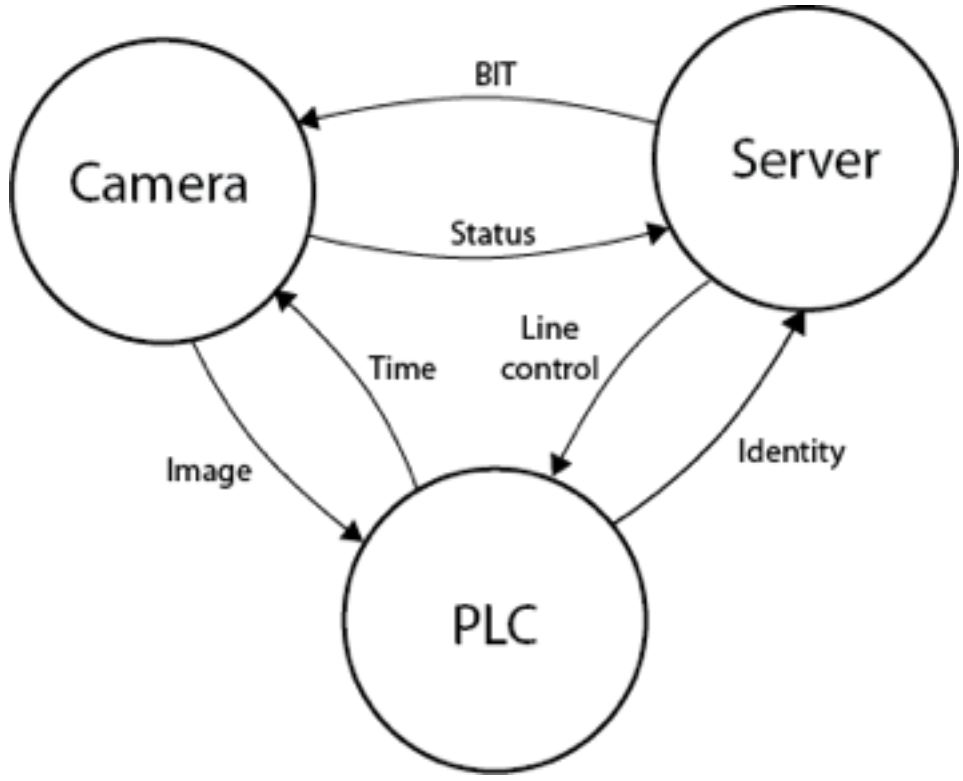
Chapter 13

Figures

FIGURES OFTEN CONTAIN AMBIGUITIES

Figures can be used together with requirements. Figures can also be a part of the rationale for the chapter. Because pictures can be open for interpretation, it is probably better to associate the figure with a rationale than a requirement. It is strongly suggested that a defined notation be used, for example UML or ERD (Entity Relationship Diagram). Own notations can easily be misunderstood, where rectangles, circles and arrows are used without any proper definition. If the figure is referenced from within the requirement, it must be as clear and unambiguous as other requirements.

Figure 13.1 Figures can be open for interpretation.



Chapter 14

Subordinate Clauses

SUBORDINATE CLAUSES SIDE-TRACK READERS

A tagged requirement consists of one or several grammatically complete sentences. Each complete sentence consists of a main clause and zero or more subordinate-clauses. The following sentence contains a main clause and a sub-clause.

> The system logs the time, when the user logs in.

The main clause is 'The system logs the time". The subordinate clause is "when the user logs in". The main clause can form a complete sentence by itself; it makes sense to say: "The system logs the time". But the sub-clause can not form a complete sentence on its own; it does not make sense to only say: "When the user logs in". The sub-clause needs the main clause.

It is very common to use subordinate clauses in running text. They give the reading variety, which is good for keeping the interest awake. But they also side-track a reader from the main subject, which can cause misunderstandings. Complicated sentences that contain several subordinate clauses are not desirable from a clarity point of view.

The following text is an exaggeration to show what frequent use of sub- clauses can do to the understanding of a text.

> The Application checks the user name, such as "admin", and password, which must contain between 4 and 10 characters, including at least one numeric, in order to prevent unauthorized access, even if a perpetrator can still physically access the system hardware.

If the author doesn't speak the language like a native, it is very likely that the author will use subordinate clauses incorrectly. Due to these reasons, requirements should

contain as few subordinate clauses as possible. It is better to construct a main clause from the sub-clause and add a new sentence to the requirement. The following requirement...

AS-2-C	The Accounting system contains one object type, which is Marketing.

... can easily be rewritten into two sentences. And we have gained some clarity.

AS-2-D	The Accounting system contains one object type. The object type is Marketing.

The rule is to eliminate subordinate clauses as much as possible, and rewrite them into complete sentences.

But there are sub-clauses that we need for a requirement to work. These clauses are called *temporal clauses* and *clauses of condition*.

TEMPORAL SUB-CLAUSES

Temporal sub-clauses begins with:

- After
- As
- As soon as
- As long as
- Before
- Directly
- Immediately
- The moment
- Once
- Since
- Until
- When
- Whenever
- While

XB-73-C	After the user has opened an XML-document, the XML parser builds a tree structure from the elements in the document.

The temporal sub-clause is a condition for the main clause to be true. If the temporal clause is not true, the complete requirement can not be fulfilled.

DM-264-A When the user pushes the store button, the Database Manager stores the XML document into the database.

The temporal clause can also be placed after the main clause.

CONDITIONAL SUB-CLAUSES

Requirements can not exist without conditional sub-clauses. Conditional sub- clauses are the most frequently used sub-clauses in any Requirements Specification. Conditional sub-clauses begins with:

- If
- Unless
- In case
- In the event that
- On condition that
- Provided that
- As long as

For example:

MT-370-A If the measurement falls outside the boundaries, the Metrics Generator displays the measurement in red.

Another example:

AS-334-C Provided that a Company has been connected to the discount plan, the Company receives the following monthly volume discount for all of the Company's broadband service subscriptions.

Table 3.

FROM AMOUNT (EURO)	TO AMOUNT (EURO)	DISCOUNT (%)
0,00	1499,99	0 %
1500,00	2499,99	15 %
2500,00	4999,99	17 %
5000,00	9999,99	19 %
10000,00	14999,99	21 %
15000,00		23 %

It is interesting to compare this type of construct with ordinary programming languages, for example C:

```
if (measurement < minBoundary || measurement > maxBoundary) {
     DisplayInRed(measurement);
}
```

The similarity is remarkable. In fact, programming languages are a subset of natural languages (English), and we can use this to our advantage when writing requirements. If we think in terms of programming language constructs (if-statements, while-statements, until-statements) our requirements get clearer and are less likely to be misunderstood by a reader. An experienced writer of requirements uses patterns like this. It may be boring to read, but the Requirements Specification is not intended

to entertain its audience. If the author still has literary ambitions, this can be expressed in the Rationale or the introductory chapters.

Other sub-clauses

The general rule is to avoid using other types of sub-clauses. The author should rewrite the sub-clause into a main clause (a new sentence) if possible.

RELATIVE SUB-CLAUSES BEGINS WITH

- If
- Unless
- In case
- In the event that
- On condition that
- Provided that
- As long as

SUB-CLAUSES OF REASON OR CAUSE BEGINS WITH

- because
- inasmuch as
- since
- as
- for

SUB-CLAUSES OF PURPOSE BEGINS WITH

- so that
- in order to
- so as to
- for the purpose of
- to

CONCESSIVE SUB-CLAUSES BEGINS WITH

- though
- although
- even if
- while
- whereas
- as

SUB-CLAUSES OF RESULT BEGINS WITH

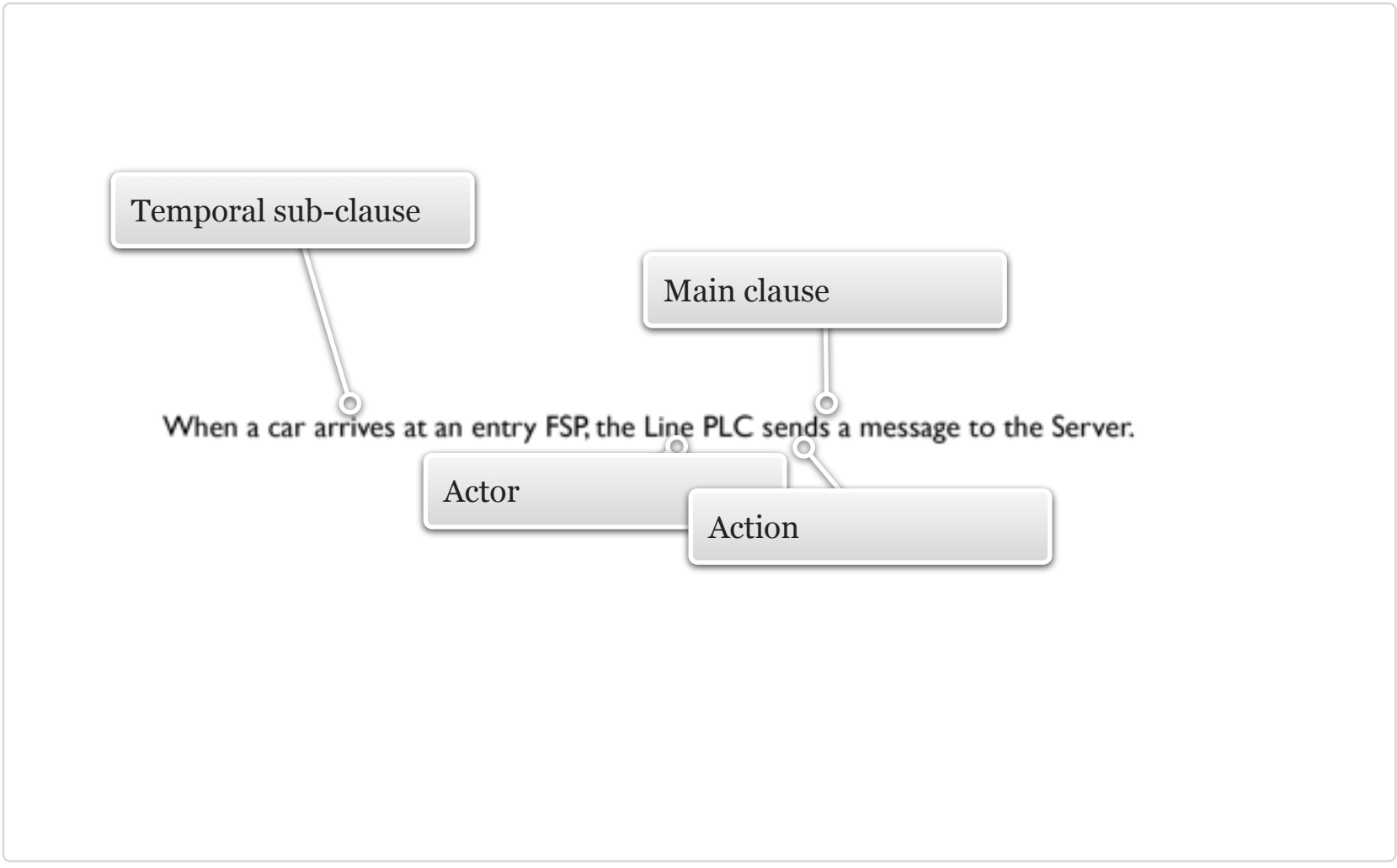
- so that
- in order that

SUB-CLAUSES OF COMPARISON BEGINS WITH

- as
- like
- the way
- as if
- as though

The relative sub-clause is commonly used when writing requirements, but can almost always be rewritten into a new sentence. Using words like "which" and "who" can lead to misunderstandings and should be replaced by its proper noun ("Metrics generator", "invoice", "user" etc.).

INTERACTIVE 14.1 The anatomy of a requirement in natural language.



Chapter 15

Word Classes

ADJECTIVES CAN BE PRECARIOUS

Systems are best described by what they do and by what properties they possess. System actions are for the most parts verbs (looking from a grammatical perspective). The core of the action is a verb. A system is poorly described through adjectives, because adjectives are widely open for interpretation. Words like accurate, good, high, profitable, even, fast, satisfactory, maximal, standard, able etc., do not describe a system from any point of view. Moreover, these types of words are not testable, and it can not be decided whether a test case passed or failed based on these words.

But there are exceptions, like the words "empty" or "last". There is not much to discuss if something is empty, or something occurs last in a sequence of events.

If it is safe to use an adjective is often decided by the context (the rest of the requirement). On the whole, adjectives are best used by Marketing executives to convince customers to buy the system.

ADVERBS CAN BE EVEN MORE PRECARIOUS

Adverbs are as precarious to use as adjectives. In fact, they are more problematic. Adverbs are often used to "exaggerate" (or modify) other words, for example: very, extremely, nearly, highly, evenly, specifically, accurately, exactly, especially... The list is long. In reality, adverbs bring little information when it comes to describing a system – and what it brings is open for interpretation. Adverbs are easy to identify, because they often end in "-ly".

But there are exceptions even when it comes to this rule. The following adverbs may be used with caution:

- never
- here
- where
- therefore
- thereby
- thereof
- there
- latest

The following adverbs may also be used with caution:

- daily
- hourly
- weekly
- monthly
- quarterly
- yearly

NOUNS ARE SELDOM PROBLEMATIC

Nouns can describe people (e.g. operator, customer, subscriber), things (e.g. Invoice, Accounting system, file), properties, states and events (e.g. correctness, quickness, answer). Concrete nouns (people and things) can be directly observed by our external senses and are seldom problematic to use when writing requirements. They are rarely misunderstood, although not exactly described. Abstract nouns on the other hand can only indirectly, through their effects, be observed by our external senses. Abstract nouns such as "correctness", "quickness", and "answer" can be difficult to use while trying to obtain clarity. Often it can help to make a definition of the word in the *Definitions* chapter or in the *Glossary* chapter, when in doubts.

NOMINALIZATIONS CAN BE ABSTRACT

Nominalizations are verbs and adjectives that have been converted into nouns, for example *output* from *put out*, *requirement* from *require*, and *failure* from *fail*. Even the word *nominalization* is in itself a nominalization.

Too much nominalization in a document can sound abstract and be difficult to understand. A Requirements Specification contains often several hundred nominalizations. It is not possible to remove them all, and while a requirement text needs to be as clear as possible, a plain text paragraph can afford to contain nominalizations. Do not, however, convert nominalized nouns back to their original adjective.

Nominalizations are recognized by words that ends in:

- –ment
- –ance
- –ence
- –ion

Chapter 16

Special expressions

Using special expressions in requirements

The English language contains lots of constructs that does not suit requirements very well. For example:

- and so forth
- and so on
- etc.
- i.e. (ie)
- e.g. (eg)
- for example
- at least

Avoid these types of constructs.

Chapter 17

Pronouns

PRONOUNS ACT AS REFERENCES

Pronouns (I, you, it, any one) are often used as references in a text. There are many types of pronouns:

- Personal pronouns
- Possessive pronouns
- Reflexive pronouns
- Demonstrative pronouns
- Relative and interrogative pronouns
- Exclamative pronouns
- Expressions of inexact quantities

Pronouns can reference to people or things, it can be used to mark possessions, it can reference to things that are nearby in space or time, or link a clause to another clause etc.

It is common to avoid certain types of pronouns in technical documents, such as personal pronouns, reflexive pronouns and possessive pronouns. It makes the document “impersonal” and possibly more objective. Instruction manuals (such as this book) can use some pronouns to make it more personal for the reader.

Because pronouns often reference people, things or other clauses in a text, they are subjected to ambiguity. Using the pronoun *it* in a requirement text leaves it up to the reader to decide if *it* means User, Metrics generator or Invoice.

Reading a Requirements Specification, it is very common to find requirements on the form:

> AS-112-B It shall be possible to...

Omitting the *actor* in the requirement leaves it to the reader to sort it out who is doing this.

PERSONAL PRONOUNS

Personal pronouns should never be used in any type of technical document. Personal pronouns can be used in instruction manuals and similar publications. The following personal pronouns exists:

- I, me
- you
- he, him
- she, her
- it
- we, us
- they, them

REFLEXIVE PRONOUNS

Reflexive pronoun are used to refer back to the subject of the clause or sentence and should be avoided.

- myself
- yourself
- himself
- herself
- itself
- oneself
- thyself
- ourselves
- yourselves
- themselves

POSSESSIVE PRONOUNS

Possessive pronouns are acting as markers of possession and defines who owns a particular object or person.

- my, mine
- your, yours
- his
- he, hers
- its
- our, ours
- their, theirs

DEMONSTRATIVE PRONOUNS

Demonstrative pronouns refer to things that are either nearby in space or time, or to things that are farther away in space or time. Avoid demonstrative pronouns in requirement texts.

- this, these
- that, those
- the one, the ones
- any one
- whoever
- such
- same
- former
- latter
- ones

RELATIVE AND INTERROGATIVE PRONOUNS

Relative pronoun are used to link one phrase or clause to another phrase or clause. Interrogative pronouns are used in questions, and should never occur in a requirement text.

- who, whose
- which
- whom
- that
- what
- whoever
- whichever
- whatever
- whenever
- wherever
- however

EXCLAMATIVE PRONOUNS

Exclamative pronouns are used in exclamative sentences, for example: “Some packets may contain errors.”

- some
- no
- any
- somebody
- someone
- anybody
- anyone
- nobody
- no one
- none
- something
- anything
- nothing
- neither
- either
- nowhere
- noplace, no place
- somewhere
- anywhere
- anyplace
- someplace
- somewhat
- somehow
- sometime

EXPRESSIONS OF INEXACT QUANTITIES

Expressions of inexact quantities are exactly what they say, they are inexact and should not be written in a requirement text. There can be exceptions, but the rule is generally true.

- all
- every
- each
- most
- much
- many
- a great deal of
- a great many
- a large amount of
- a large number of
- a lot of
- lots of
- plenty of
- a little, little
- several
- a few, few
- a couple of
- both
- one
- everybody
- everyone, every one
- everything
- everywhere
- everyplace, every place
- everyday
- large part of
- majority of
- a good deal of
- a good many
- a certain amount
- a number of
- a considerable part
- a considerable portion
- part of
- less
- east
- fewer
- fewest
- another
- other, others
- else
- elsewhere

Chapter 18

Other guidelines

THE LENGTH OF A SENTENCE

Do not use more than 20 words in a sentence. A requirement can off course consists of more than one sentence. If a sentence becomes too long, try and rewrite it into two or more sentences.

REPEAT KEY NOUNS

Do not be afraid to repeat key nouns in a requirement text. If the *DHCP server* is the key noun in a requirement, write *DHCP server* as often as necessary. Never use pronouns to refer to key nouns.

SETTING THE CONTEXT FOR A REQUIREMENT

Sometimes the author includes "this means ..." constructs in their requirements.

> IF-53-B The IP-filter processes the incoming TCP packets. This means that the IP-filter decides if a TCP packet should be passed or blocked depending on the filtering rule.

The writer has simply failed to explain what the requirement really means. Instead, the writer can modify the first sentence into a chapter heading, for example:

> 5.2.6 Processing of incoming TCP packets

The heading (together with the rationale) gives the context for a requirement, and the information is therefore not needed in the requirement itself. A reader knows that all requirements in the current chapter concerns incoming TCP packets, and not outgoing.

IF-53-C	If the TCP packet fulfills the filtering rule, the IP-filter passes the TCP packet.
IF-54-A	If the TCP packet violates the filtering rule, the IP-filter blocks the TCP packet.

SENTENCES SHOULD COMMUNICATE ONE IDEA

Authors may feel the need to include as much information into the same sentence as possible:

AS-366-A	Itemized data may be created as a text file, which is located in the directory $ACCOUNT_HOME/account_files/invoices/itemized in UNIX.

Requirement AS-366-A uses both the passive voice, and a relative sub-clause (two violations). The author can easily rewrite the requirement by using two sentences.

AS-366-B	The accounting system operator can create itemized data as a text file. The accounting system writes the text file to the UNIX directory $ACCOUNT_HOME/account_files/invoices/itemized.

WRITERS THAT ARE AFRAID TO COMMIT

In TS-58-A the author fails to put into words what the author really meant. The second part of the sentence doesn't communicate any information of interest. Some authors may do this on purpose, as they are afraid to commit to a decision. Instead, the writer should list all 7 travel zones and their zone names in a table, as in requirement TS-328-A.

TS-58-A There are 7 different travel zones, defined according to their zone names in Gothenburg.

If the author does not know, or is afraid to commit, there is no point in writing detailed requirements. The author should instead spend some time trying to resolve all open issues. There is no point in writing a fuzzy Requirements Specification only to satisfy the customer or the Quality Assurance standard, and then postpone decision making until the implementation phase. Problems should be highlighted and dealt with as early as possible, unless there are specific reasons not to. Dealing with key problems in mid-implementation is usually a bad strategy.

Chapter 19

Document layout

PRESENTING REQUIREMENTS TO A READER

Requirements are mostly written in documents that are stored electronically and occasionally printed as hard copies. But even if requirements are stored in a database, or written as an XML-document, they need to be presented to a reader in the best possible way.

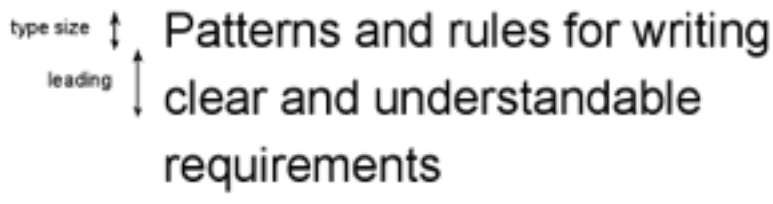
When a writer works with layout, the writer should first decide if the document will be read as a hard copy or mostly viewed on-screen. This influences the design of the document. The writer may decide that the document must work both ways and a compromise is therefore necessary.

Textual material should occupy about 50 percent of a page, the other 50 percent is margin, header and footer. The writer should use left justification for all texts. Pictures, tables and lists should not be centered.

The line length is between 40 and 70 characters for hard copy reading, and between 40 and 60 characters for on-screen viewing. This is equal to 8–12 words per line. If the writer is using a sans serif type, the line should be 8–9 words, while a serif type can contain 10+ words.

The writer can use two different fonts in the document, usually one for the headings (sans serif) and one for the body text (serif). Sans serif is easier to read on the screen, and the writer may therefore use this for on-screen viewing of body texts. If the writer selects serif for body texts, sans serif can be used for captions and headings.

The top ten list for sans serif fonts are: Helvetica, Univers, Frutiger, Futura, Franklin Gothic, Optima, Gill Sans, Avante Garde, and Myriad. This book uses Avenir Next Condense and Helvetica Neue as headings, and Georgia as body text.



The difference between type size and leading.

Body text font sizes are 9, 10 or 11 points for hard copies. On-screen viewing may be as large as 15, 16 or 17 points. Leading is the amount of vertical space between lines of types. Typesetters (the real persons) used to insert one or more thin strips of lead between lines to make the text more readable, thus the term “leading”. A body text that has a 9, 10 or 11 font size should use between 1 and 3 points of space between lines. A 12 point font size needs 2–4 points of space, a 14 point font size needs 3–6 points, a 16 point font size needs 4–6 point, and a 18 point font size needs 5–6 points of space. Leading is expressed as two numbers: the first is the typeface’s point size, and the second is the baseline-to-baseline measurement, for example 9/12, 12/16, 16/21 and so on. Foot notes uses 8/9 points. Body texts need more leading and headings need less leading. The body text in this book uses a 17/20.4 leading, but this can be changed in the reading application.

To emphasis words in the text, italic is used. Do not use bold or underlined to emphasis words. Italic is however harder to read on-screen. Headings can off-course use bold with advantage.

Use a 12 points leading between paragraphs. The leading between the heading and the body text should also be 12 points. You can have both leading and indent between paragraphs, but if extra space is inserted between paragraphs, there is no need for indentation. Indents are used to separate paragraphs. The first paragraph after a heading has therefore no need of indentation.

Bulleted lists have 2–3 characters between the bullet and the text. The same is true for numbered lists.

Do not use the heading “Introduction” and similar boilerplates. Headings should say something about the content of the chapter.

Chapter 20

Conclusions

REWRITE REQUIREMENTS

Regardless of all patterns and rules, there is really only one proven method to get clear and understandable requirements: rewrite, rewrite and rewrite. Requirements need to be read with a critical eye and rewritten, more than once. Authors in general are, however, too happy too soon with their work. The first draft of a requirement is not seldom the final version. Good authors, on the other hand, are seldom satisfied with their first draft.

Hopefully, this guide will help software engineers to develop a critical eye when reading requirements. Having a critical eye is half the work, rewriting is the rest.

Bibliography

- Svartvik, Jan and Sager, Olof, “Engelsk universitetsgrammatik”, Stockholm: Liber AB, 1996.
- Schriver, Karen A. “Dynamics in document design”, New York: Wiley Computer Publishing, 1997.
- Koren, Leonard & Meckler, R. Wippo, “Graphic Design Cookbook”, San Francisco: Chronicle Books, 1989.
- Keller, Arnold “The Practical Technical Writer”, New York: Pearson Longman, 2004.