

**SE 2AA4, CS 2ME3 (Introduction to Software
Development)**

Winter 2018

26 Specification Quality

Dr. Spencer Smith

Faculty of Engineering, McMaster University

March 8, 2018



26 Specification Quality

- Administrative details
- Line Formatter Example
- Converting English to Mathematics

Administrative Details

- A3
 - ▶ Part 1 - Specification: due 11:59 pm Mar 12
 - ▶ Part 2 - Code: due 11:59 pm Mar 26
- A4
 - ▶ Your own design and specification
 - ▶ Due April 9 at 11:59 pm

Line Formatter Overview

- The [line formatter specification](#) was written to improve on existing specifications
- How well does the specification do with respect to the following qualities: abstract, correct, unambiguous, complete, consistent and verifiable?
- For a requirement specification like that given, what are the advantages and disadvantages of maintaining both a formal specification and a natural language specification?
- A relevant critique in: [Meyer \(1985\)](#) “On Formalism in Specification”

Line Formatter

- **Input** stream signalled with ET
- Exactly one ET character in each input stream
- **Character** classifications:
 - ▶ Break character - BL (blank) and NL (new line)
 - ▶ Non Break Character - all others except ET
 - ▶ End of text indicator ET
- **Word** is a non-empty sequence of non break characters
- **Break** is a sequence of one or more break characters
- **Output** same sequence of words, except if there is an oversize word
 - ▶ Oversize means more than MAXPOS characters, where MAXPOS is a positive integer
 - ▶ If there is an oversize word
 - ▶ Set Alarm to TRUE
 - ▶ Exit the program

Line Formatter

- Up to the point of an error, the program's output should have the following properties
 - ▶ A new line should start only between words and at the beginning of the output text, if any
 - ▶ A break in the input is reduced to a single break character in the output
 - ▶ As many words as possible should be placed on each line (i.e. between successive NL characters)
 - ▶ No line may contain more than MAXPOS characters (words and BLs)

Abstract?

Abstract?

- Not abstract!
- Specifies an implementation for error handling (variable named Alarm)
- Do not have to name the variable Alarm
- Could use exception handling (or another approach) instead
- ET is a machine dependent (program domain) concept

Correct?

Correct?

- The definition of line is incorrect!
- A line is defined as being between NLs, which ignores text before the first NL and after the last NL
- The output file does not contain ET, which is either a bug in the spec or a significant non-uniformity

Unambiguous?

Unambiguous?

- Ambiguous!
- “point of error” is not defined
- Output matches input to the last acceptable word, or the last acceptable character?
- “trailing blanks ending with ET” is ambiguous
- The program’s output should be the same sequence of words as in the input
 - ▶ But the input is not a sequence of words
 - ▶ If the input were a sequence of words, what about leading or trailing breaks?
 - ▶ “As many words as possible should be placed on each line”
 - ▶ WHO WHAT “NL” WHEN
 - ▶ WHO “NL” WHAT WHEN

Complete?

Complete?

- Not complete!
- Meaning of NL and its relation to the concept of line is left implicit
- Alarm is not specified if MAXPOS is never exceeded

Consistent?

Consistent?

- Not consistent!
- “non-empty” and “one or more” (synonyms)
- “stream” and “sequence” (synonyms)
- Is the input a “stream of characters” or a “sequence of words separated by breaks”? – sequence of T is not the same as sequence of sequence of T

Verifiable?

Verifiable?

- The specification cannot be verified, since it is ambiguous and incorrect

Advantages and Disadvantages?

- Advantages and disadvantages of maintaining both formal and a natural language spec?

Advantages and Disadvantages?

- Advantages and disadvantages of maintaining both formal and a natural language spec?
- Advantage of natural language - understandability
- Advantage of formal spec
 - ▶ Unambiguous
 - ▶ Highlights difficult to informally detect cases
 - ▶ Checking for completeness and consistency
 - ▶ Amenable to tool support
- Advantage of both - all of the above advantages
- Disadvantages - have to maintain two specs
- Automatic translation
 - ▶ Formal spec to natural language has been researched
 - ▶ Natural language to formal spec has received more attention

English and Mathematics as Languages

- English is a language
- So is Mathematics
- Both have
 - ▶ Rules of grammar (syntax)
 - ▶ Semantics
- When writing in any language, pay attention to grammar and semantics. Get both right.

English and Mathematics: A Difference

- In English and other natural languages
 - ▶ Ambiguity desired, intentionally possible
 - ▶ Unambiguous statements almost impossible
- In Mathematics
 - ▶ Ambiguity not desired, intentionally prevented
 - ▶ Ambiguous statements almost impossible (even in probability theory, fuzzy logic)

Mathematics and Engineering

- Therefore, mathematics is the language of engineering

Correct Syntax for Mathematics

- Make sure the syntax of your mathematical expressions is correct
- Correct syntax does not guarantee correct semantics
- Incorrect syntax makes the mathematics ambiguous
- Example problems to watch for
 - ▶ Mismatch of types
 - ▶ $(p.xcoord() + width) \wedge p.xcoord()$
 - ▶ $\neg(\text{integer})$
 - ▶ Set of sequences accessed by $[i]$
 - ▶ $\forall(i : \mathbb{N} \wedge j : \mathbb{N} \dots)$
 - ▶ Use of programming language notation in mathematics
 - ▶ Integer values instead of boolean
 - ▶ `length == MAX_SIZE`
 - ▶ $(x > 7) = \text{true}$ instead of $(x > 7)$ (bad form)

Different World Views

- English and other natural languages
 - ▶ Express both static and dynamic views
 - ▶ States and actions (verbs of being and action)
- Imperative programming languages
 - ▶ Primarily dynamic world view (changes)
- Functional programming languages
 - ▶ Static world view
- Mathematics
 - ▶ Static world view only
- Fundamental conceptual differences

Static Versus Dynamic Views

- These very different world views pose a conceptual hurdle for the translator
- The translator must bridge the gap between
 - ▶ Dynamic and static view of problem statement
 - ▶ Dynamic world view of programming and
 - ▶ Purely static world view of mathematics
- Not hard, but requires conscious attention

Translating Between Languages

- Translating a statement from one language to another is a multistep (not single) process
 1. Statement in source language to a mental understanding of the meaning of the statement
 2. Reformulate mental understanding into target language view, concepts, culture
 3. Mental understanding of the meaning of the statement to a statement in the target language
- The first and last statement must mean the same

Translators

- Knowing two languages: not enough to translate
- A good translator knows well
 - ▶ The two languages
 - ▶ AND the subject being translated
 - ▶ AND how to translate
- These three things are different

Organization and Style

- When writing in English or any other natural language, one pays careful attention to
 - ▶ Organization of the essay, report, etc.
 - ▶ Style of expression
- When writing in Mathematics, to do the same:
 - ▶ Clear, complete, concise
 - ▶ KISS (Keep it Simple Sharp and Straightforward)
 - ▶ Understandable
 - ▶ Interesting

Strategies

- Understand the meaning of the original
- Obtain all needed information
- Close the gap between the English text and mathematics
- Divide and conquer (complexity)

Strategy: Understand the Original

- Describe specific instance of general problem
- Distinguish essentials from background
- Draw a diagram
- Express in intermediate or mixed language
- Identify object referred to
- Identify implicit (but false) “information”
- Identify missing information
- Identify relationships between essential objects
- Identify special cases

Strategy: Obtain all Needed Information

- Ask the author of the task description
- Identify gaps in the description of the task
- Identify implicit “information”
- Ask if implicit “information” may be assumed
- Identify data present and ask about related details
- Ask if missing information is really needed
- Read **carefully, thoroughly, precisely**

Strategy: Close Gap English – Mathematics

- Express implicit information explicitly
- Reduce vagueness and ambiguity
- Reword English text to be closer to mathematics (express in an intermediate, mixed language)

Strategy: Divide and Conquer

- Construct a table
- Distinguish between specific cases
- Introduce an auxiliary mathematics function
- Modularize

Strategy: Draw Diagrams, Describe Specific Instances of the Given Problem

- Graphical representations help understand the meaning of the message
- For specific instances, think of extreme cases first to simplify
 - ▶ $n = 0$
 - ▶ $n = 1$
 - ▶ $n = \text{inf}$
- Think of a normal sized problem, usually something like $n \geq 3$
- You might want to write down truth tables

A Small Translator's Glossary

English	Mathematics
and, but	\wedge
or	\vee
for all, each, every, any	\forall , \wedge series, universal quantification
for no, none	\forall , \wedge series, universal quantification, with a negated assertion
there is (are), there exist(s), for some, at least one	\exists , \vee series, existential quantification

A Small Translator's Glossary Continued

English	Mathematics
integer	$\dots \in \mathbb{Z}$
sorted	$\wedge_{i=0}^{n-2} A[i] \leq A[i+1],$ $\forall (i : \mathbb{N} 0 \leq i < n - 1 : A[i] \leq A[i+1])$
if (when, whenever) ... then ...	$\dots \rightarrow \dots$, sometimes \wedge
search, find, equal, present	$=$
exchange, rearrange, different order, different sequence, merge, copy, sort	permutation

Your Translator's Glossary

- A professional translator compiles his/her own translation glossary
 - ▶ Over time
 - ▶ Based on own accumulated experience
- You should too!

Exercise

Consider an array D with index values ranging from 1 to n . The subject of this example is part of a specification for a subprogram that will count how many times a particular given value occurs in the array D .

The goal of this exercise is to write a postcondition for the subprogram, relating the various relevant variables values when the search is complete.

Exercise Continued

Understand the task in the original language

- Identify objects referred to (look for nouns in the original English text)

- Identify missing information

Exercise Continued

Understand the task in the original language

- Identify objects referred to (look for nouns in the original English text)
- Array D , index value, times (count), particular given value, relevant variables's value
- Identify missing information

Exercise Continued

Understand the task in the original language

- Identify objects referred to (look for nouns in the original English text)
- Array D , index value, times (count), particular given value, relevant variables's value
- Identify missing information
- Names of variable for: index, times (count), particular given value

Exercise Continued

Understand the task in the original language

- Identify objects referred to (look for nouns in the original English text)
- Array D , index value, times (count), particular given value, relevant variables's value
- Identify missing information
- Names of variable for: index, times (count), particular given value
- Are there any other relevant variables?

Exercise

- Identify missing information
- Names of variable for
 - ▶ Index: assume i
 - ▶ Times (count): ask the author of the task, assume *count*
 - ▶ Particular given value: Ask the author of the task, assume *key*
 - ▶ Are there any other relevant variables? (no?)

Reference

- Baber, Robert L., *Translating English to Mathematics*, 2002