Writing Cautions

Justin Zobel

University of Melbourne, Australia

Economy, vigour, clarity, ambiguity

- ‘The need for reduced-complexity interaction is of high potential to be widespread.’
- Unnecessary text, unnecessary words, long words, waffle.
- Purpose of text is not thought out, or the text is obsolete.
- Inadequate revision: reluctance to self-criticize, listen to critics, try out rewrites, or edit at all.
- Informality or matiness.

And don’t overqualify . . .

- Claims made without adequate qualification: replace ‘users are’ by ‘users can be’.
- Excessive caution: replace ‘might be possible’ by ‘is possible’.
- Double negatives. ‘Not necessarily unaffected’?

Sloppiness

Sloppy titles:

- Choose the right keywords.
- Should be short but informative.

Ragged, run-on sentences:

- Have a simple structure, with a single thought or topic.
- Thoroughly revised.
- Not ‘sentences, like this one, which I hate to encounter in a thesis because I know they will be impossible to correct, sometimes seeming to arise from lack of confidence, where a writer isnt quite sure what she or he wants to say, or may even have lost track of what they want to say, and so says several things in the one sentence that might almost be contradictory; and sometimes arise from overconfidence, where the writer genuinely has a complex concept to communicate to the reader and tries to discharge the whole explanation in a single sentence, and the effect is the same, namely, a confused mess with excessive, or even absurd, punctuation, and a strangled syntax that no likely reader will be able to digest, if they even get that far.’

Brevity, lack of examples

- Make abstract ideas concrete – people learn by generalization from instances.
- Use them. If you have some examples, add more.
- The system stores ‘document records’. What does a typical record look like?
- The communication module ‘has a simple but inflexible command syntax’. What input would be incorrect?
- No: ‘Process ownership failure can lead to corruption of the lock tables.’
- Yes: ‘Process ownership failure, for example when the TCP/IP connection is broken and communication with the client is lost, can lead to corruption of the lock tables.’

Similarly, making the background too brief:

- Many writers expect far too much of their readers.
- All assumptions should be made explicit.
- Write for yourself – as you were the day you commenced your project.

Jargon

- Technical terminology is needed for discussion of technical concepts.
- Technical terminology excludes less informed readers.
- Dense technical writing is hard to read.
  'Dependence of the meaning of the vague-world assumption on a specific extension, and implicitly the intension, is a result of integrating exclusion set with capture-negation semantics'?
  'Most interpreter-oriented parsers recursively construct operator trees via single-function unit procedures and path failure on invalid syntax, with sandbox housekeeping, whereas the iterative parser in SCRAM interleaves parsing and execution, using ad hoc exception management and explicit memory recovery trails'?

Pomposity

Supposedly ‘scientific’ writing:

- ‘The execution of the algorithm is such that it completes in an unusually short space of time.’
- ‘The algorithm is fast.’

Consistency and flow

Poor motivation, flow, organization:

- Explain the purpose, value, importance of each element (section, theorem, algorithm, definition).
- Link adjacent material so that it flows.

Lack of consistency:

- In notation, mathematics, terminology.
- In layout, format.
- In captions, figures, tables.

Historical fallacies

- ‘Since the invention of the internet, researchers have been using the web to publish data.”
  - The web was invented in 1990 and only became generally available within academia in 1993–4.
  - The internet was invented in 1969 and became a primary information medium for computer scientists and physicists during the 1980s.
- Decisions that look poor in retrospect may have made perfect sense at the time. (In the context of hardware limitations, known algorithms, accepted results, ...)

Likewise, nonsense, absurdity, over-generalization:

- ‘Execution was almost instantanious.’
- ‘The web is effectively infinite.’ (Unbounded? – maybe.)
- ‘There’s no limit to the possible efficiency gains.’

Ugly mathematics

- Choose variable names wisely.
- Use tools that can typeset mathematics to textbook standard; then use them correctly.
- Don’t overuse or overload subscripts.
- Spend time designing the layout of anything complex – use pen and paper.
- Don’t number things unnecessarily.
- Explain it thoroughly. Mathematics is not a substitute for text.
- Stay within your competence; don’t show off.

Meaningless algorithms

Readers need to understand why an algorithm is interesting, and why they should believe it is correct.

Expect to include most of:

- The steps of the algorithm.
- Input, output, and internal data structures.
- A proof of correctness.
- Complexity analysis.
- Experiments that confirm its behaviour.

Meaningless algorithms . . .

When evaluating performance, consider:

- The basis of evaluation (for example, functionality or efficiency, type or properties of data).
- Resource requirements.
- Applicability.
- Matching theoretical predictions and experimental results.

Unreadable algorithms

Use informal language; write
`for each character` c `in string` s, not
`for` $1 \leq i \leq |s|$`, let` $c \leftarrow s[i]$.

Discuss each step; explain its purpose as well as its function.

Use mathematical notation rather than programming notation. Sets are more convenient than lists; write x_i, not `x[i]`; write $x \times y$ or xy, not `x * y`.

Avoid ridiculous detail; don't use a loop to compute a sum.

Give enough information – a reader should not have to reinvent your algorithm to implement it.

Unreadable algorithms . . .

Is it described for a computer or a person?

```
ctr = 0;
s = readln(input);
while eof(input)!=true
   for each i, 1<=i<=strlen(s)
      if s[i]=='A' then ctr++;
   s = readln(input);
```

1. Initialise by setting $ctr \leftarrow 0$.
2. For each string *s* in the input,
   - 2.1 Let *a* be the number of 'A's in *s*.
   - 2.2 Add *a* to *ctr*.

Unreadable algorithms . . .

Who is the algorithm being explained to:

- A programmer who has to implement it?
- A reader who has to understand the method?

Questions that an explanation needs to address:

- Is the algorithm plausible? Is it feasible?
- Is there enough detail to allow implementation?
- Is it clear that the algorithm actually solves the problem in question? What problem is it intended to solve?
- What other literature is there that might be considered? Is there a relevant web resource that a reader should consult?

Uninterpreted experiments

- Describe the data, the testbed, the parameters.
- Explain the results for the reader.
- Give the arguments connecting the evidence to the hypothesis.
- Include both detailed analysis of each individual experiment and a global summary of what the results show.

Unexplained data

- Where did it come from?
- What are its limits and failings?
- What are the units? What do the numbers mean?
- How much is there?
- Is it simulated (artificial) or real?
- If artificial, why is it ‘like’ real data?

Unreadable tables

STATISTICS	SMALL	LARGE
Characters	18,621	1,231,109
Words	2,060	173,145
After stopping	1,200	98,234
Index size	1.31 Kb	109.0 Kb

Characteristic	Collection	
	Small	Large
File size (Kb)	18.2	1,202.3
Index size (Kb)	1.3	109.0
Number of words	2,060	173,145
After stopping	1,200	98,234

Table guidelines

Columns should consist of values of the same kind (although not necessarily of the same type).

Think of rows as tuples, columns as attributes.

Make use of table hierarchy, to group together things that belong together.

Avoid vertical rules, excessive horizontal rules.

Be selective, don't put in too much information.

Align numbers on the decimal point.

All tables should have the same orientation.

Useless figures

Too many figures look dreadful – badly designed, confusing, congested, unprincipled, meaningless lines and shapes.

The caption and picture should be reasonably self-sufficient: explain units, features, main purpose.

Illustrate one important idea or result.

Use graphs in preference to tables.

Only illustrate significant features – be selective.

Don't include a graph just because some software generated the data. Any illustration should have a clear role in the narrative.