Intro to Methods of Software Engineering Fall 2016

Lecture 5 — October 13, 2016

Patrick Lam *version 1*

Paragraphs

Paragraphs help readers understand your text better by making its structure more obvious. A wall-of-text, by contrast, is hard to read, and many readers will just give up and move on to the next thing.

Each paragraph should contain one message. Use an appropriate number of sentences to convey the message. Each sentence should contribute some new information, while showing a connection to the previous sentence. Aim to make your paragraphs self-contained, that is, meaningful in isolation; provide some context.

The best paragraphs succinctly state their message near the beginning, ideally in the first sentence. (That is, however, not a hard-and-fast rule). Texts that follow this rule are easier to skim, since it's possible to just look at the first sentence.

Some examples of paragraphs from *Trees, maps, and theorems* by Jean-luc Doumont:

BAD (misleading lede):

> Single-use, disposable medical devices are packaged and sterilized by the manufacturer. Their packaging must provide protection, facilitate sterilization, maintain sterility... Reusable devices, by contrast, must be ...

GOOD:

> Medical devices may be broadly divided into two categories, disposable and reusable, having different sterilization requirements. Single-use, disposable devices are packaged and sterilized by the manufacturer... Reusable devices, by contrast, must be ...

BAD (rambling storytime!):

> Figure 2 shows the evolution of the Ge content in the SiGe layer. Obviously there is a nearly linear decrease of the Ge content with increasing fluence. Knowing the ...

GOOD:

> The germanium content decreases linearly with increasing fluence (Figure 2).

Written Communication: Code and Emails

Code documentation

Some best practices for code comments:

1. structure your code (just like your writing).
   (avoid 2000-line methods!)

2. choose good names for variables, methods/functions, classes.

   - `i` is totally fine for a loop variable!
   - follow language conventions, e.g. `getX()` versus `x()` (or just allowing direct access to `x`).
   - describe what the thing is/does, e.g. `exceptionQueue()`.

3. in your code, add comments when there are complicated passages (better yet, simplify the code!). One exception: "I tried the obvious thing X but it fails because Y" is a good comment.

4. use good parameter names and include documentation comments for methods and classes, including `@link`/`@see` references whem appropriate.

```
/**
 * A computational engine for solving relational satisfiability problems.
 * Such a problem is described by a {@link kodkod.ast.Formula formula} in
 * first order relational logic; finite {@link kodkod.instance.Bounds bounds} on
 * the value of each {@link Relation relation} constrained by the formula; and
 * a set of {@link kodkod.engine.config.Options options} specifying, among other global
 * parameters, the length of bitvectors that describe the meaning of
 * {@link IntExpression integer expressions} in the given formula.   The options are
 * usually reused between invocations to the {@linkplain #solve(Formula, Bounds) solve}
 * methods, so they are specified as a part of the {@linkplain KodkodSolver solver} state.
 *
 * <p>
 * A {@link KodkodSolver} takes as input a relational problem and produces a
 * satisfying model or an {@link Instance instance} of it, if one exists.  Some
 * implementation of this interface can also produce a {@link Proof proof} of
 * unsatisfiability, if the given problem has no models.
 * </p>
 *
 * @specfield options: Options
 * @author Emina Torlak
 */
```

5. use unit tests and assertions (more later).

Writing emails

Reading and writing emails is central to my job. Although writing code might be more important for some of you, writing good emails will boost your effectiveness. Here's some email-writing tips.

Let's consider a sample email.

```
To: Stéphane Somé <ssome@site.uottawa.ca> ①
From: Patrick Lam <se-director@uwaterloo.ca>
Subject: chemistry & CEAB accreditation ②
Date: Wed, 5 Oct 2016 13:26:34 -0400

Hi Stephane,

③

I've heard that your Software Engineering programme managed to get
re-accredited without mandatory Chemistry. We've been trying to
figure out what's the smallest amount of chemistry we could include
here at Waterloo in Software Engineering.

Did you receive any feedback from the CEAB about not including
   chemistry? ④

Thanks!

Patrick Lam
Director, Software Engineering Programme ⑤
University of Waterloo
```

Some important points:

1. recipient: make sure you're sending your email to the right person (in this case, I had to do a couple of Google searches); use "cc" for those who need to know but not act.
2. subject line: super important! empty subject lines often get ignored; good subject lines provide context and summarize actions required.
3. provide context: you might judiciously quote the preceding message or otherwise explain the context.
4. make the request: be explicit and say what action you require the recipient to take;
5. signature (as relevant)