Some tips on good writing
Part I: Basic composition

R. E. K. Stirewalt, Laura K. Dillon
Software Engineering and Network Systems Lab
Michigan State University

Writing is a design problem

Chief obstacle to good technical writing is inherent complexity of ideas
As with programming, you manage complexity by introducing structure (i.e., modules)
  - Modules exist at many levels of abstraction
  - Chapters, sections, sub-sections, and even paragraphs
As with programming, there are many criteria for decomposing documents into modules
  - E.g., structural analysis (separate inputs, transformers, and outputs)
  - E.g., information hiding and stepwise refinement

Example: Amalia paper

We had to cover lots of ideas:
  - Software componentry
  - Design for extension/contraction
  - Structural operational semantics
  - Formal specification languages
  - Mixin layers
  - Class models
  - Software generation
  - Proof of correctness via transparency
  - Software integration
Motivate -> describe -> reflect

Readers want these concerns separated:
- Want to understand the motivation for a solution w/o getting into the details of how it works
- Want to understand the mechanism w/o getting sidetracked by why it’s so cool or its long-term implications
- Want to see a careful analysis that reflects on the whole solution

Gotta have all three, or you’re not done!

MDR and use of formal methods

When a solution exploits a formal method, you can an alternative to MDR pattern:
- Brief introduction to notation [motivation]
- Description of your solution [description]
- Concrete example to illustrate how it works [reflection]

Pattern is particularly useful if you need to invent (or specialize) some notation
As with MDR, if you don’t have all 3, you aren’t done!

MDR and 5-paragraph intros

Introduction attempts to convey a key idea, which is what you ultimately want to describe
1. Introduction to problem, culminating in thesis statement
2. Why is this problem interesting and hard [motivation]
3. Key idea that supports your thesis [description]
4. Validation and properties of your solution [reflection]
5. Restatement of thesis and roadmap
**Subordinate your ideas**

Each module should contain one key idea, which might be supported by many subordinate ideas (i.e., sub-modules).

Hierarchical structure of the modules should reflect the subordination structure of the ideas.

This is particularly true when the module is a section or a paragraph.

---

**Subordination in a paragraph**

Each paragraph should encapsulate one (and only one) thought.

Topic sentence expresses this thought.

Other sentences develop and/or elaborate (i.e., support) this thought.

---

**Define and consistently use terms**

Define every technical term that you use:

- Emphasize the term with its definition.
- Never use a term in a context that requires the reader to know the meaning of a term before you define it.
- Never re-define a term or add further constraints on its meaning later in the paper (called *remorse*).

Don’t use multiple terms for the same idea.
Example: How to use a term before defining it

In the introduction of [DiSt'03]

... For example, in [1] we argue that the visitor pattern [4] is the key to integration and that GenVoca generators [5] are a key enabling technology for supporting extension and contraction. ...

Observe:

- Several terms “used” w/o defin
- Why is that OK?

How not to use term before defining it (example 1)

Also in an introduction:

“...Target language features are generated when inference rules conclude. Because none of the inference rules were able to conclude, no target-language features were generated, and thus no translation was taking place; our inference rules defined a meaningless function.”