Computer Science MS Project

Final Report\textsuperscript{1}

\textsuperscript{1}Adapted from slides by Joe Geigel.
Overview

Guidelines for final report

▶ Important Dates and Resources
▶ Content and structure
▶ Format
Final Report

Important dates:

- Oct. 2 (Fri): Outline (tentative) and Introduction (draft)
- Oct. 26 (Mon): Additional Section (draft; for peer review)
- Nov. 20 (Fri): Full Draft to faculty advisor (recommended)
- Dec. 2 (Wed): Final Draft to CS Dept.
Final Report

Discuss final report with your faculty advisor early and often:

- Ask for exemplar project reports supervised by your advisor
- Ask about expectations and requirements (length, particular content, . . .)
- Share outline and drafts and ask for feedback

Recall that final report is most significant component of course grade:

- 45% assigned by faculty advisor
Final Report

What (does a good project final report look like)

- Consider the research papers that you have read
- Ask for exemplar project reports supervised by your advisor
- Look at previous CS MS Projects, especially “Best Paper Award” recipients:
  - Answering Why-Not Queries Through Modification-based Explanations; Hiteshi Satish Shah (2181; Dr. Carlos Rivero)
  - Music Style Transformer: Generation and Synthesis of Music via Raw Audio Transcription with Applications to Style Transfer; Jason Palmer St George (2185; Dr. Hans-Peter Bischof)
  - Pattern-Aware Compression and Query of Log Files; Eric Hartman (2195; Dr. Carlos Rivero)
  - Automated Feedback tool for Computing Theory; Aniruddha Shukla (2195; Dr. Ivona Bezakova)
  - Multi-Instrument Audio to Midi Prediction; Jack Spencer Smith (2195; Dr. Hans-Peter Bischof)

Ask yourself why these papers/reports are good?
Final Report

How (do you write a good project final report)

- Research, Writing, and Presenting section of Resources page
  (https://www.cs.rit.edu/~mtf/teaching/20201/msp/resources.html)
  has some presentations on these topics.

- Writing for Computer Science (3rd edition) by Justin Zobel
Final Report

Why (is writing a good project final report difficult)

The main cause of incomprehensible prose is the difficulty of imagining what it's like for someone else not to know something that you know.

S. Pinker, *The Sense of Style*

- Must think of writing from the perspective of the reader
- You already know everything, your reader does not yet know anything
  - How did you get to know everything (about your project)?
  - What is the story that you are trying to tell?

A cornerstone of good writing is identifying what the reader needs to learn. A strong thesis or paper has a story-like flow, with a sequence of concepts building from a foundation of knowledge assumed to be common to all readers up to new ideas and results. Thus an effective paper educates its readers. It leads readers from what they already know to new knowledge you want them to learn.

J. Zobel, *Writing for Computer Science, Ch. 3*
Scientific Method and Engineering Design

**Scientific Method**

- Ask a Question
  - Do Background Research
  - Construct a Hypothesis
  - Test with an Experiment
  - Procedure Working?
    - No
    - Yes
      - Analyze Data and Draw Conclusions
        - Results Align with Hypothesis
          - Communicate Results
        - Results Align Partially or Not at All with Hypothesis
          - Troubleshoot procedure. Carefully check all steps and set-up.

- Experimental data informs background research for new/future project. Ask new questions, form new hypotheses, experiment again.

**Engineering Method**

- Define the Problem
  - Do Background Research
  - Specify Requirements
  - Brainstorm, evaluate, and choose solution
  - Develop and prototype solution
  - Test Solution
    - Solution Meets Requirements
    - Communication
    - Solution Meets Requirements Partially or Not at All
      - Communicate Results
      - Based on results and data, make design changes, prototype, test again, and review new data.
Always communicate results!
Academic/Technical Papers

Most papers follow a structure that (somewhat) aligns with scientific method and/or engineering design:

▶ Abstract
▶ Introduction
▶ Background
▶ Body (most variation)
  ▶ Hypothesis / Requirements
  ▶ Evidence / Description
  ▶ Analyze Data / Evaluate Solution
▶ Conclusions
▶ Future Work
▶ Acknowledgements
▶ References
▶ Appendices (much more common in MS Capstone reports)

*The written work rests on a program of activity that begins with interesting questions and proceeds through a sound methodology to clear results.*

*J. Zobel, Writing for Computer Science, Ch. 3*
Abstract / Introduction

Abstract:
► 1 - 2 paragraphs
► a concise summary of paper’s topic, approach, and results
► *not* the first paragraph(s) of the introduction

Introduction:
► 2 - 4 pages
► an expanded version of the abstract
► *but* does not flow from the abstract

More people will read the abstract and introduction than any other section.

*The opening paragraphs can set the readers attitude to the whole paper or thesis, so begin well. All of a document should be created and edited with care, but take the most care with the opening, to create the best possible first impression. The abstract should be written especially well, without an unnecessary word, and the opening sentence should be direct and straightforward. . . . That is, describe what you have done without the details of how it was done.*

J. Zobel, *Writing for Computer Science, Ch. 7*
Abstract / Introduction

Every paper answers a question or solves a problem

► Describe what the question / problem is
► Describe why the question / problem is important
► Describe (at a high-level) the contribution:
  ► core idea that is novel
  ► methodology employed to validate

A preview of the main body and conclusions of the paper.

CGI model (Derek Dreyer)

► Context: motivate the general topic
► Gap: explain specific problem and why existing work does not solve it
► Innovation: state what is new and explain how it fills the gap

A “top-down” approach: general to specific / general to expert
Abstract / Introduction

The Introduction sets the rules of the game:

- A good paper is often good because it delivers on its promise: “We will show XYZ” and, indeed, XYZ is shown in the paper.

- A bad paper is often bad because it fails to deliver on its promise; easily disappointed by a paper that promises much and delivers little.
Background

- No question / problem or methodology is ever brand new
- Almost all work builds on existing work
  - Reference and briefly describe this other work
  - Give credit where credit is due

- Sometimes includes a longer, more detailed description of the what and why of the question / problem
  - All the details that the reader needs to know to understand the contribution
  - Might introduce a motivating example

- Sometimes includes a detailed analysis of related work, especially if this work directly builds on previous paper(s) or if this work directly addresses limitations of previous paper(s)
  - Interestingly, in my sub-field (Programming Languages), it is most common to have an explicit Related Work section immediately before the Conclusions section.
Tell the story of your work.

_The body of a good paper — everything between the introduction and the conclusions — should have a logical flow that has the feel of a narrative._

_J. Zobel, Writing for Computer Science, Ch. 3_

Because every project is different, the body of each paper is a little different. Usually comprised of three or more sections, each telling a different chapter of the story.

How you present your project is different from how you executed it.
Body

Tell the story of your work.

Because every project is different, the body of each paper is a little different. Usually comprised of three or more sections, each telling a different chapter of the story.

How you present your project is different from how you executed it.

Many different organization strategies. Typically:

▶ the “key ideas”
▶ followed by “technical meat”
▶ followed by “evidence and analysis”

In “quotes”, because never actually title sections with these terms.
Body

Methods of structuring the body:

- As a Chain
- By Specificity
- By Example
- By Complexity
- Key Ideas / Technical Details (Derek Dreyer)
Methods of structuring the body:

- **As a Chain**
  Linear, logical flow from background to results:
  explain problem, review previous solutions, describe the new solution, demonstrate improvement

- **By Specificity**
- **By Example**
- **By Complexity**
- **Key Ideas / Technical Details (Derek Dreyer)**
Body

Methods of structuring the body:

▶ As a Chain
▶ By Specificity
  Outline the high-level, then fill in the details.
  For example, a new algorithm comprised of a number of sub-routines.
▶ By Example
▶ By Complexity
▶ Key Ideas / Technical Details (Derek Dreyer)
Methods of structuring the body:

- As a Chain
- By Specificity
- By Example
  Introduce idea or result with concrete examples, then explain formally. For example, when describing a new language feature first give example programs written with feature, then describe the feature in detail (maybe with additional examples to motivate interesting aspects).
- By Complexity
- Key Ideas / Technical Details (Derek Dreyer)
Body

Methods of structuring the body:

- As a Chain
- By Specificity
- By Example
- By Complexity
  Start with a simple case, then proceed to a more complex case.
  For example, in a distributed systems context, start with the assumption that all links and machines are identical, then proceed to handle the case when they differ.
- Key Ideas / Technical Details (Derek Dreyer)
Body

Methods of structuring the body:

- As a Chain
- By Specificity
- By Example
- By Complexity
- Key Ideas / Technical Details (Derek Dreyer)
  - Key Ideas: use high-level intuition to describe the “interesting” takeaway
  - Technical Details: use the scaffolding of “Key Ideas” to develop general solution
Body

Methods of structuring the body:

- As a Chain
- By Specificity
- By Example
- By Complexity
- Key Ideas / Technical Details (Derek Dreyer)

These methods can and should also be used to structure individual sections and sub-sections of the body. For example, whole body may be organized as a chain, but the description of the new solution might be organized by complexity.
Conclusions / Future Work

Conclusions

▶ Based on the analysis / evaluation, what can be concluded:
 ▶ Was the hypothesis confirmed? Did the solution work?
 ▶ Be honest about any limitations with analysis / evaluation.

Future Work

▶ No work is ever complete:
  ▶ What planned work was not completed (and why).
  ▶ New questions / problems identified (especially by negative results).
  ▶ Limitations that could be addressed.
  ▶ Invitation to others (and yourself) to continue the work.
Acknowledgements / References / Appendices

Acknowledgements

- Thank the supporters of the work (including funding agencies)

References

- Proper citation of work is essential

Appendices (much more common in MS Capstone reports)

- Additional supporting evidence that would detract from main narrative:
  - proofs of supporting lemmas
  - per-benchmark execution time statistics
  - full details of user surveys (e.g., included text or images)
Academic/Technical Papers

Most papers follow a structure that (somewhat) aligns with scientific method and/or engineering design:

▶ Abstract
▶ Introduction
▶ Background
▶ Body (most variation)
  ▶ Hypothesis / Requirements
  ▶ Evidence / Description
  ▶ Analyze Data / Evaluate Solution
▶ Conclusions
▶ Future Work
▶ Acknowledgements
▶ References
▶ Appendices (much more common in MS Capstone reports)

_The written work rests on a program of activity that begins with interesting questions and proceeds through a sound methodology to clear results._

_J. Zobel, Writing for Computer Science, Ch. 3_
Low-Level Mechanics

- **Flow** (within paper, within section, within paragraph):
  Should be clear how each sentence relates to the adjacent ones. Begin with old info, end with new info.

- **Coherence** (within paper, within section, within paragraph):
  Should be clear how each sentence relates to the big picture. “One paragraph, one point” — each paragraph serves one purpose.

- **Consistency**:
  Lots of variation between papers on capitalization of section titles, intra-document references (e.g., “Fig. 3” vs. “Figure 3”), spelling convention (e.g., American “color” vs. British “colour”). Make a choice, but apply it consistently throughout report.

- **Definition before Use**:
  Be sure to define a term, variable, acronym the first time it is used, but maintain flow and coherence.
Low-Level Mechanics

- Style (*WfCS* Chs. 6-7)
- Punctuation (*WfCS* Ch. 8)
- Mathematics (*WfCS* Ch. 9)
  - Key theorems and lemmas should be named and referenced by name (rather than by number).
- Algorithms (*WfCS* Ch. 10)
  - Give types of inputs, output, and variables.
- Graphs, Figures, and Tables (*WfCS* Ch. 11)
  - Line plots are overused; be sure that connecting distinct outputs is meaningful.
  - Avoid double y-axes; can suggest false relationships between the two axes by adjusting the ranges.
- Code
  - Do not include long listings of source code.
  - But, illustrative examples are appropriate.
Low-Level Mechanics

- Citations are parentheticals, not nouns:
  - [17] shows that computing 0CFA via least-fixed iteration is $O(n^5)$.
  - ✔ Computing 0CFA via least-fixed iteration is $O(n^5)$ [17].

- References should be complete and authoritative:
Academic Integrity and Plagiarism

Your MS Capstone Final Report (and Poster) must be *your* original words.

- Using text (or figures) from papers or websites without proper acknowledgement is a serious offense with serious consequences. 
  (No different from “cheating” by using someone’s homework.)

- Submitted final reports will be evaluated with *turnitin.com*.

- Does not mean that it is never appropriate to quote; simply do so with proper attribution and sparingly.
More Tips

Final report will be the permanent record of your capstone work:

- Should be significant (i.e., lengthy).
- Should be clear and complete.
- Find a balance between failing to sufficiently describe the accomplished work and describing every aspect of the work in pedantic detail.
- Should be professional.
- Should be a guide to others aiming to extend the work.
- Share drafts with peers for feedback and proofreading.
Format

All CS MS Capstone Final Reports must use a provided template:
https://www.cs.rit.edu/usr/local/pub/GraduateProjects/capstone_template.zip

- Microsoft Word (discouraged)
- \LaTeX \document preparation system (recommended)
  - Markup language with separation of content from presentation, like HTML and CSS
  - Available for all platforms (https://www.latex-project.org/get/): TexLive (Linux), MacTeX (MacOS), MikTeX (Windows), Overleaf (online)
  - Excellent support for mathematics, intra-document references, and bibliographies (BibTeX)
  - Will offer a brief (optional) tutorial on Monday
Schedule

- Friday (no class)
- Friday (Oct. 2 @ 8:00pm): Final Report - Outline and Introduction due
- Monday (optional class): \LaTeX{} tutorial
- Wed/Fri (Week 8) and Mon/Wed (Week 9): Milestone 2 Presentations
  - Wednesday (Oct. 7 @ 8:00AM): Milestone 2 Presentation due
- Friday (Oct. 9): Milestone 2 Deliverables due to Faculty Advisor
Final Report - Outline and Introduction

Due Friday (Oct. 2) @ 8:00PM

► Outline (tentative)
  ► PDF format required; file should be named report_outline.pdf
  ► CS MS Project template not required
  ► A trivial “Introduction, Background, Methodology, Evaluation, Conclusion” outline will receive 0 credit;
    it should be possible to identify the project from the outline.
  ► Be as detailed as possible,
    with planned sections, sub-sections, sub-sub-sections (as necessary), figures, tables, definitions, theorems, examples, etc.
Final Report - Outline and Introduction

Due Friday (Oct. 2) @ 8:00PM

▶ Introduction (draft)
  ▶ PDF format required; file should be named report_intro.pdf
  ▶ Minimum 1000 words (approx. 2 full columns of the CS MS Project template)
  ▶ CS MS Project template not required
  ▶ Write as complete a draft as possible for the “Introduction” section.
  ▶ Assume the audience has a general understanding of computer science, but not expertise in the area of the project.
  ▶ Low-level formatting (fonts, margins, etc.) is not a primary concern; focus on writing some text to get feedback.
Milestone 2 Presentation

Due Wednesday (Oct. 7) @ 8:00AM

- Milestone 2 Presentation
  - PDF format required; file should be named milestone2.pdf
  - 6.5 minutes (+ 1.5 minutes questions)
  - For a general computer science audience, but one familiar with your previous Project Description presentation.
  - Title: Project title, student, faculty advisor
  - Brief (1 slide, 30 seconds) summary / reminder of project
  - Brief (1 slide, 30 seconds) summary / reminder of accomplishments thus far
  - Summary of Milestone 2
    - Goal: What was originally planned (activities and/or deliverables) for Milestone 2?
    - Progress: What has been accomplished towards each Milestone 2 component? Describe any substantial difficulties or changes.
    - Next step: What is the most important task to be addressed next?
  - Expectation is that you are presenting work in progress and a status report.