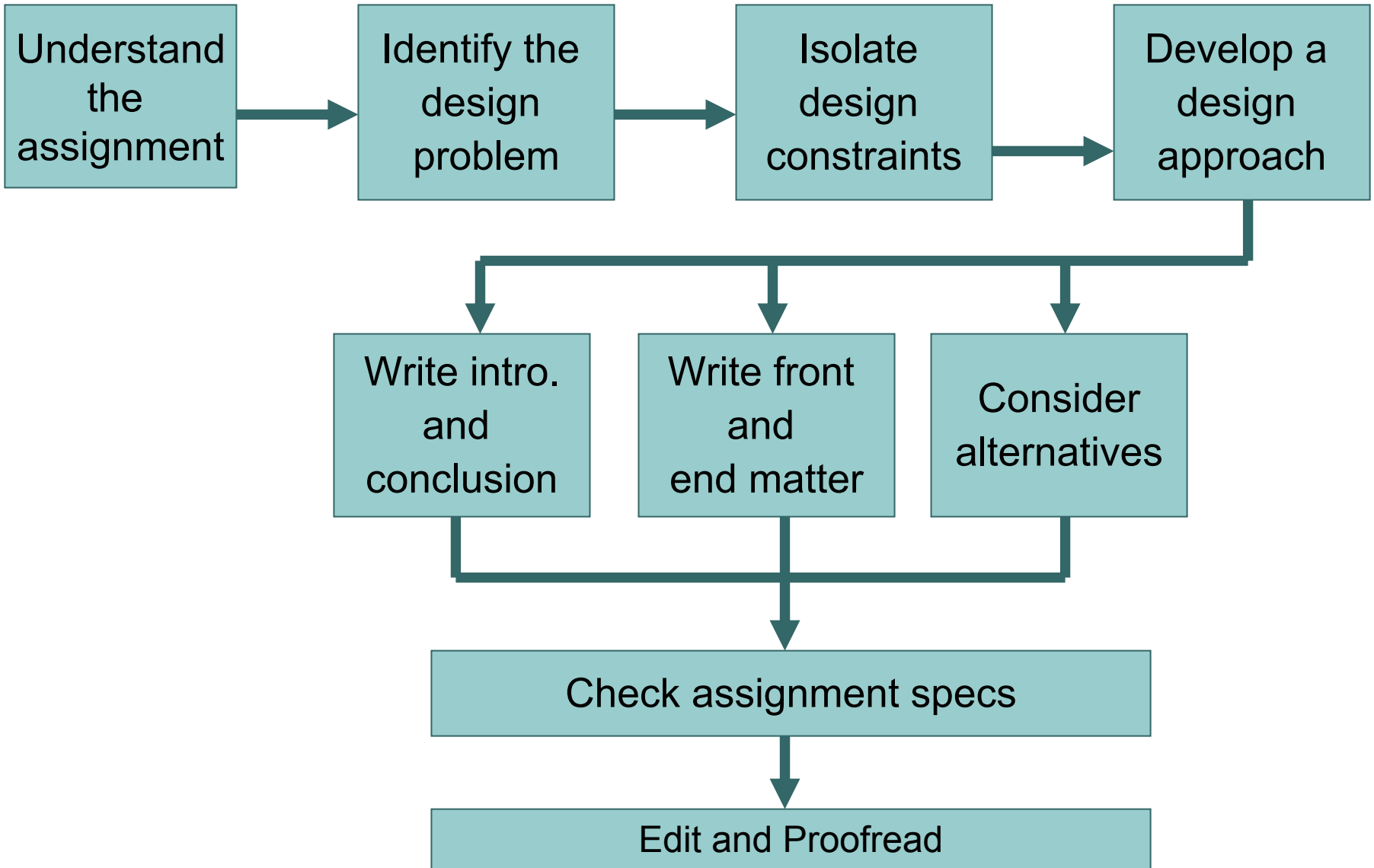
How to Write a Design Report for 6.033

Mya Poe
MIT Program in Writing and Humanistic Studies
March 14, 2003
myapoe@mit.edu

9 Steps in the Writing Process

1. Read and understand the assignment
2. Identify the design problem
3. Isolate specific design constraints
4. Develop a design approach
5. Consider alternatives
6. Write your introduction and conclusion
7. Write your front and end matter
8. Check assignment specs
9. Edit and proofread

9 Steps in the Writing Process



Step #1

Read and understand the assignment

- Underline important design concepts
- Identify report deliverables

III. Resource limitations and design goals

Because the mote is so small, it has very strict resource limitations in terms of energy. Your design will have to cope with the tradeoffs introduced by these limitations. We also want to make as many of the motes (which are disposable) as possible: **your design must balance reducing the cost of the mote with improving performance and saving energy.** . .

We estimate that a good paper will be 8 to 10 pages in length (single spaced).

Step #2

Identify the design problem

- In one sentence, what design problem are you solving?
  - The problem statement helps you clarify the purpose of your project by making you focus on the “bigger” picture.

The problem I am trying to solve in this project is _____.

The purpose of this project is to _____.

Step #3

Isolate design constraints

- What limitations will you consider in your design?
  - Make a checklist using the assignment sheet, or
  - Number the constraints on the assignment sheet

Examples:

√ 16-bit microprocessor (running at 8Mhz)

√ 32Kbytes of RAM (with a bandwidth of 8Mbs)

√ “You are only concerned with interactions between programs”

Step #4

Develop a design approach

- Begin with a **conceptual overview** of your design
  - You may need a figure to clarify your ideas
- Follow with a detailed **design description**:
  - Move from general to specific
  - Use section headings to help readers understand hierarchy of ideas
  - Chunk information into readable units
  - Use figures, tables, and pseudo-code to illustrate concepts
- **Tip:** Do not throw away alternative design ideas

2 Conceptual Overview of the MMU

To achieve enforced modularity in a segmented memory system, each process must be able to protect its segments from other processes as it sees fit. The O/S must therefore maintain a database of permission information describing which processes are allowed to read from, write to, or execute in each memory segment. The kernel must update this permissions table whenever it services a system call to allocate a new data segment, spawn a new process, or change the permissions of a segment.

The MMU works by synthesizing queries to this permissions table from three pieces of information:

1. The process making a memory request.
2. The memory segment this process wishes to access.
3. The type of access (read, write, and/or execute) the process is asking for.

-- From "Simple Sharing and Enforced Modularity Access Control in a Segmented Memory System." Jeff Bartelma. March 21, 2002

3 MMU Details and Design Decisions

The following subsections discuss the Beta MMU design in detail . . .

3.1 Process-Based vs. Segment-Based Access Control

To design an MMU capable of enforcing modularity in every case, it is necessary to distinguish process-based access control from segment-based access control. . . If the MMU had data specifying only that code in segment X should be allowed access segment C (segment-based access control), it would fail to stop instance B of MATLAB from illegally accessing A's private data. See **Figure 1**. To prevent this problem, the MMU must control access to segments based on the process, not segment, that generated the memory request.

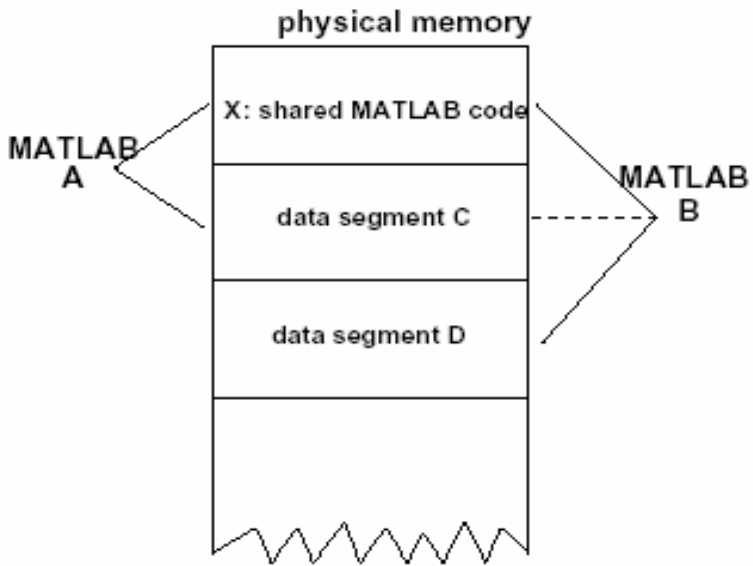


Figure 1: Two processes sharing a segment of executable code. The dotted line indicates unauthorized sharing allowed by segment-based access control.

Step #5

Consider alternatives

- Write a description of design alternatives not chosen.
  - Refer to design constraints to isolate specific variables
  - Explain the advantages and disadvantages of alternatives

  - **Tip:** No need for more than 1 page in length.

Step #6

Write introduction & conclusion

Introduction

- Explain your **motivation** for the design
  - Reference the problem you are addressing
- State your **approach** to the problem
- List specific **design considerations**
- State the **purpose of your document**

Conclusion

- Summarize your project

1 A Segmented Memory System

Most virtual memory systems provide each program module with a distinct address space, tied to physical memory that is invisible to other modules. By restricting each module's view of memory, this “standard model" of virtual memory provides excellent fault isolation. However, to claim the advantage of modularity, a programmer adopting the standard model forfeits a convenient interface for sharing memory. Two programs attempting to access the same physical location must use the different virtual addresses specified in their respective page maps. Sharing would be simpler if all processes ran in the same address space, that is, if any program could reference any piece of data, irrespective of the program that owns it. Ideally, we would like to retain isolation while providing this palatable interface for sharing.

Motivation/ Problem

← Goal

One way to approach this problem is to use a segmented memory system: divide the single address space into a number of “segments" and interpret the b high-level bits of each CPU-issued virtual address as a segment number. The remaining bits of a virtual address are used as an offset into the segment. . . .

Approach

A good MMU must enable a program to prevent all other programs from accessing its segments. That is, the MMU must be able to enforce modularity. At the same time, the MMU must facilitate useful sharing; for example, two instances of Emacs should be able to share the same segment of executable code while maintaining distinct, private data segments. Finally, the MMU must protect the kernel and provide support for its memory-related tasks, e.g., allocating segments or starting programs from within other programs.

Design considerations

This paper presents an MMU design to support segmented memory for the 32-bit Beta processor, and demonstrates its viability in light of the constraints mentioned above.

Purpose of the report

-- From "Simple Sharing and Enforced Modularity Access Control in a Segmented Memory System." Jeff Bartelma. March 21, 2002

Conclusions

End-to-end arguments are a kind of "Occam's razor" when it comes to choosing the functions to be provided in a communication subsystem. Because the communication subsystem is frequently specified before applications that use the subsystem are known, the designer may be tempted to "help" the users by taking on more function than necessary. Awareness of end-to-end arguments can help to reduce such temptations.

It is fashionable these days to talk about "layered" communication protocols, but without clearly defined criteria for assigning functions to layers. Such layerings are desirable to enhance modularity. End-to-end arguments may be viewed as part of a set of rational principles for organizing such layered systems. We hope that our discussion will help to add substance to arguments about the "proper" layering.

--From END-TO-END ARGUMENTS IN SYSTEM DESIGN, J.H. Saltzer, D.P. Reed and D.D. Clark

Step #7

Write the front and end matter

- Title
  - Brief and descriptive, using key words
- Abstract
  - Approx. 150 words summary of your report
- Table of Contents
  - Lists section headings and page numbers
  - optional

- Acknowledgements
  - Anyone who helped you with your design
- References
  - IEEE style. How? See the ***Mayfield Handbook***
- Appendix
  - Used for supplemental material
  - Not needed to understand report

END-TO-END ARGUMENTS IN SYSTEM DESIGN

J.H. Saltzer, D.P. Reed and D.D. Clark
M.I.T. Laboratory for Computer Science

Abstract

This paper presents a design principle that helps guide placement of functions among the modules of a distributed computer system. The principle, called the end-to-end argument, suggests that functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level. Examples discussed in the paper include bit error recovery, security using encryption, duplicate message suppression, recovery from system crashes, and delivery acknowledgement. Low level mechanisms to support these functions are justified only as performance enhancements.

Contents

Acknowledgements

Many people have read and commented on an earlier draft of this paper, including David Cheriton, F.B. Schneider, and Liba Svobodova. The subject was also discussed at the ACM Workshop in Fundamentals of Distributed Computing, in Fallbrook, California during December 1980. Those comments and discussions were quite helpful in clarifying the arguments.

References

[1] F. Cavalieri, T. Ruscio, R. Tinoco, S. Benedict, C. Davis, and P. K. Vogt, "Isolation of three new avian sarcoma viruses: ASV9, ASV17, and ASV 25," Virology, vol. 143, pp.680-683, 1985.

Step #8

Review the assignment specs

- Did you answer all the questions that the assignment asked you to?

Step #9

Edit & Proofread

- Did you chunk information into expected sections?
  - Title
  - TOC (optional)
  - Abstract
  - 1.0 Introduction
  - 2.0 Design Overview
  - 3.0 Design Description
  - 4.0 Alternatives
  - 5.0 Conclusion
  - Acknowledgements
  - References
  - Appendix
- Did you number the pages?
- Are all figures and tables labeled and referenced in the text?
- Are all sources cited?
- Did you avoid:
  - naked “this”
  - “the reason is because . . ”
  - “the fact that . . .”
  - phrases
  - over-use of “I”
- Did you proofread a printed copy?

Report Format

- Your name, ID#, the name of your recitation instructor, & your section meeting time at the top of the page.
- 11 or 12 point font
- Single-spaced
- No more than 10 pages, including optional Appendix.
- Write “PHASE II” on the title page if you want report submitted to the Writing Program.

Writing Help

- Model papers on 6.033 website
- Readings in your course packet
- Writing Center : web.mit.edu/writing
- *Mayfield Handbook of Technical and Scientific Writing*
- Writing Tutor:
  - Monday 10-12 — Contact mcaulf@mit.edu for appt
  - Tuesday 10-1, 5-8 — Contact mcaulf@mit.edu for appt
  - Wednesday 5:30-7:30 — first come, first served 14N-229a