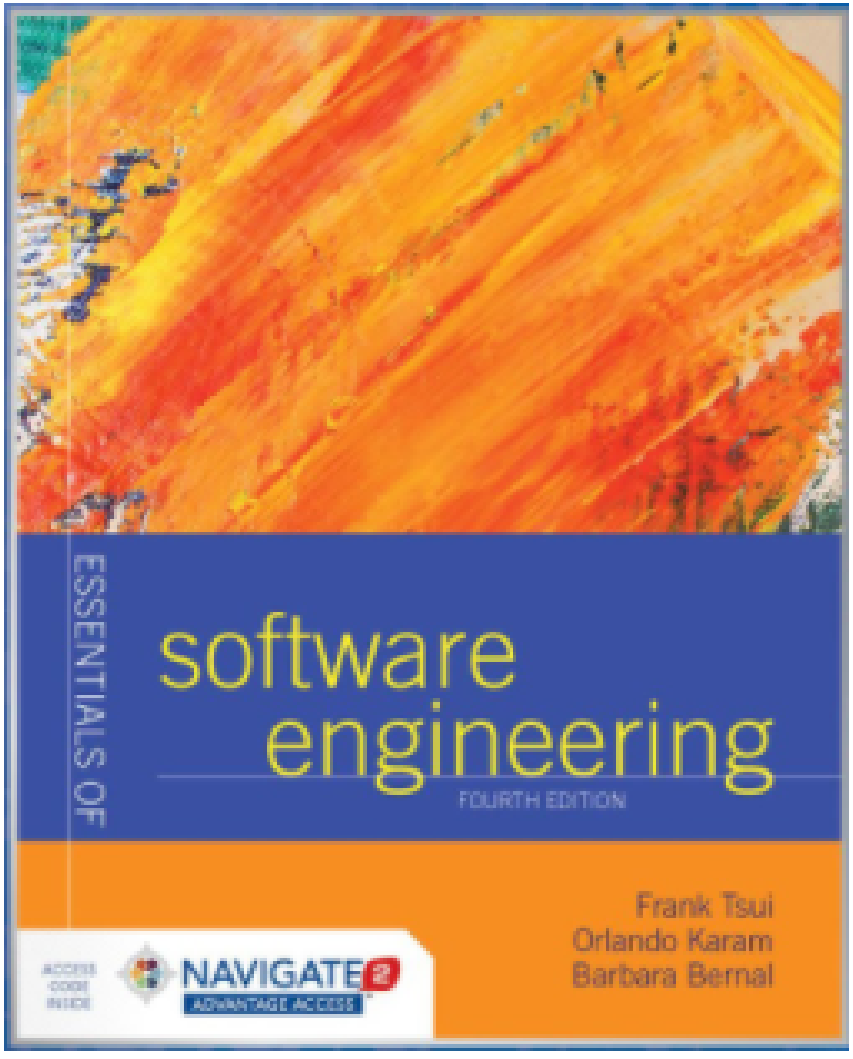
Creating a Program

CSC 354, Software Engineering I

Chapter 1: Creating a Program



Creating a Program

- We all "start" by learning how to write code in some programming language.
- Typically, with a small, hypothetical, and fairly well-defined problem.
- Usually the code in within one module.

Considerations and Decisions

- Problem Statement: "Given a collection of lines of text (strings) stored in a file, sort them into alphabetical order, and write them to another file."

Considerations and Decisions

- What are the program requirements?
  - Input formats?
  - Sorting?
  - Special cases, boundaries, and error conditions?
  - Performance?
  - Real-time?
  - Security?
  - ...

Considerations and Decisions

- What are the design constraints?
  - User interface?
  - Typical and maximum input sizes?
  - Platforms?
  - Schedule?

Creating a Program

- We learn that the program usually does not work on the first try (and probably many tries after)
- We learn about *testing* the program
- We learn about *re-reading* and *re-thinking* the (problem) requirements more carefully – then we find we may not have all the answers
- We learn about *tracing* and *debugging* the program
- Then at some point we decide that it is "good enough"

More to Consider and Decide

- Testing time
  - While the program is defined
  - While the program is developed
  - After program is completed
- Kinds of tests
  - Acceptance (validation)
  - Verification
  - Unit testing
  - Black box
  - White box

Code is "Done" – What Else Matters?

- *How long* (elapsed time) did it take to complete the work?
- *How much effort* (total person hours) is expended to do the work?
- Does the solution *solve the complete problem*?
- How "good" is the work – (code, design, documentation, testing, etc.)

How Long Does it Take?

- Recall: "write a program that reads lines from one file and writes the sorted lines to another file"
- Possible outline:
  - get the file (what language?)
  - read the file
  - sort the file alphabetically (asc? desc?)
  - write the file
  - close the file
- File contents to test: random words, special characters, blanks, different cases

How Long does it Take?

- Version 1: estimate the *ideal total time* (within 1 min); assume that you can work only on this one task, with no interruptions.
- Version 2: estimate *when you think* you will have the program done to hand over to the client
- Version 3: *Divide* the entire program into *separate developmental tasks*; these tasks might be divided into several subtasks. The current task is a planning task that has estimation as a subtask

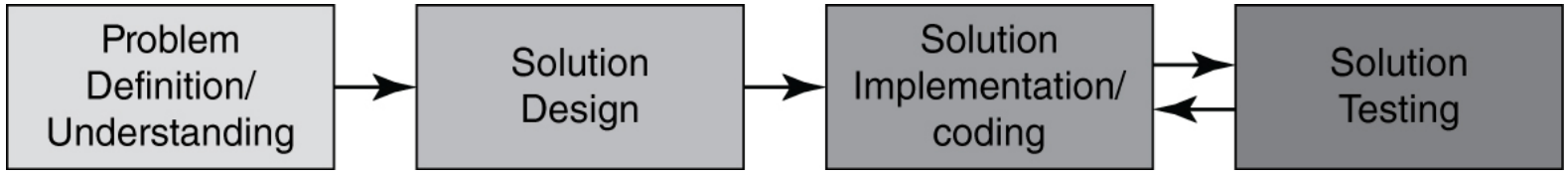
Actual Time Creating the Program

- Design and implement your solution while keeping track of the time

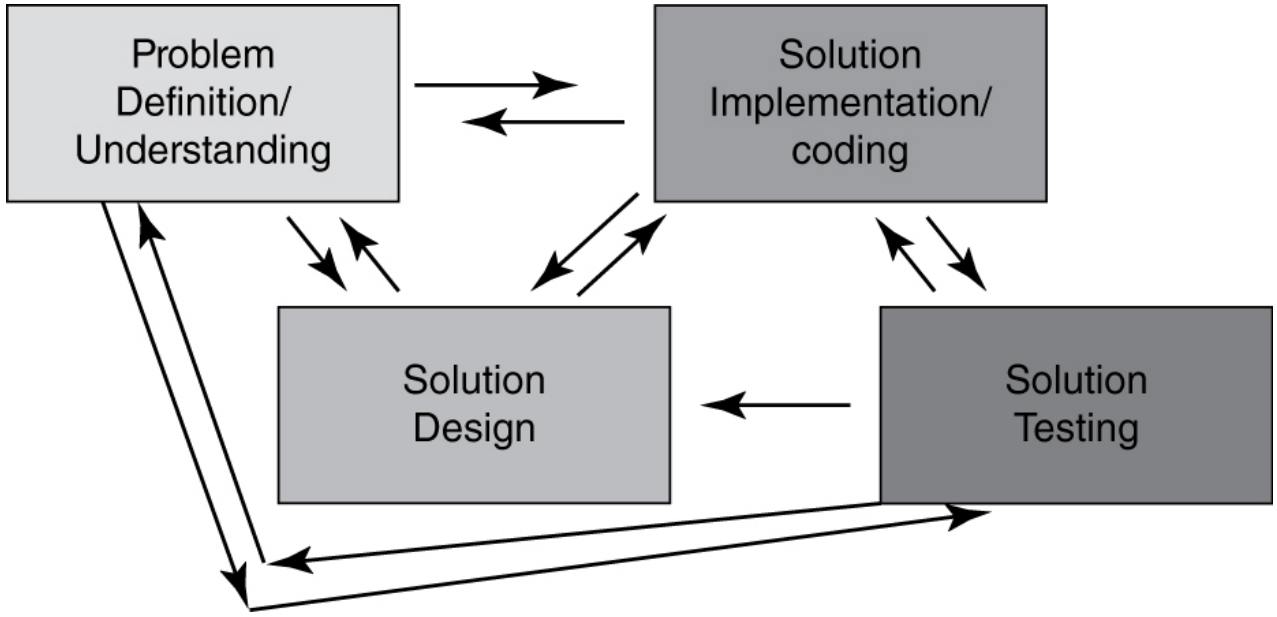
	Started	Ended	Breaks	Time
Planning				
Sort				
Read				
Write				
User interface				
Testing				
Total				

Planned versus Actual

- "Imagined" – Ideal



- "Actual" – Happening



Future Consideration: Tools