

Software Design
CSE 335, Spring 2006

Introduction

Stephen Wagner

Michigan State University

Structure of Course

- Lectures
 - Includes lots of material **not** in the readings
 - Course notes will be placed on the web
- Homeworks
 - Not collected or graded
 - Examples from homeworks used on exams and in class
- Individual projects
- Two midterm exams and a final exam

Software Design

- Requires more than just programming.
- Also includes :

Software Design

- Requires more than just programming.
- Also includes :
 - translating vague requirements into precise specifications

Software Design

- Requires more than just programming.
- Also includes :
 - translating vague requirements into precise specifications
 - conversing with costumers in terms of the application domain rather than the implementation domain

Software Design

- Requires more than just programming.
- Also includes :
 - translating vague requirements into precise specifications
 - conversing with costumers in terms of the application domain rather than the implementation domain
 - thinking and designing at several levels of abstraction according to the appropriate stage of development

Software Design

- Requires more than just programming.
- Also includes :
 - translating vague requirements into precise specifications
 - conversing with costumers in terms of the application domain rather than the implementation domain
 - thinking and designing at several levels of abstraction according to the appropriate stage of development
 - working in teams
- Must be familiar with **several** design approaches

Software Design

- Requires more than just programming.
- Also includes :
 - translating vague requirements into precise specifications
 - conversing with costumers in terms of the application domain rather than the implementation domain
 - thinking and designing at several levels of abstraction according to the appropriate stage of development
 - working in teams
- Must be familiar with **several** design approaches
- May not always be working with software

What this course covers

- Object-oriented programming.
- Heuristics for:
 - analyzing a problem that requires a software solution, and
 - designing a modular solution to the problem
- Models and notations for representing software designs and of the *design process*
- Why this is all so important

Reusability

- Software should be designed to be reusable
 - Saves effort
 - Increases maintainability
 - Increases familiarity
- Object Oriented code can be highly reusable
 - Class/template libraries

Patterns

- What if you need to write something entirely new?
 - Design reuse
 - Use familiar *patterns* that others will recognize
- In this course we will study several patterns
- The “Design Patterns” book is an excellent resource and contains many, many more patterns
- Programming language independent
 - C++, Java, C#

Question

1. What is the largest program that you have worked on?
2. For this program:
 - Approx. months/years to develop
 - Approx. lines of code (LOCs)
 - Approx. number of programmers

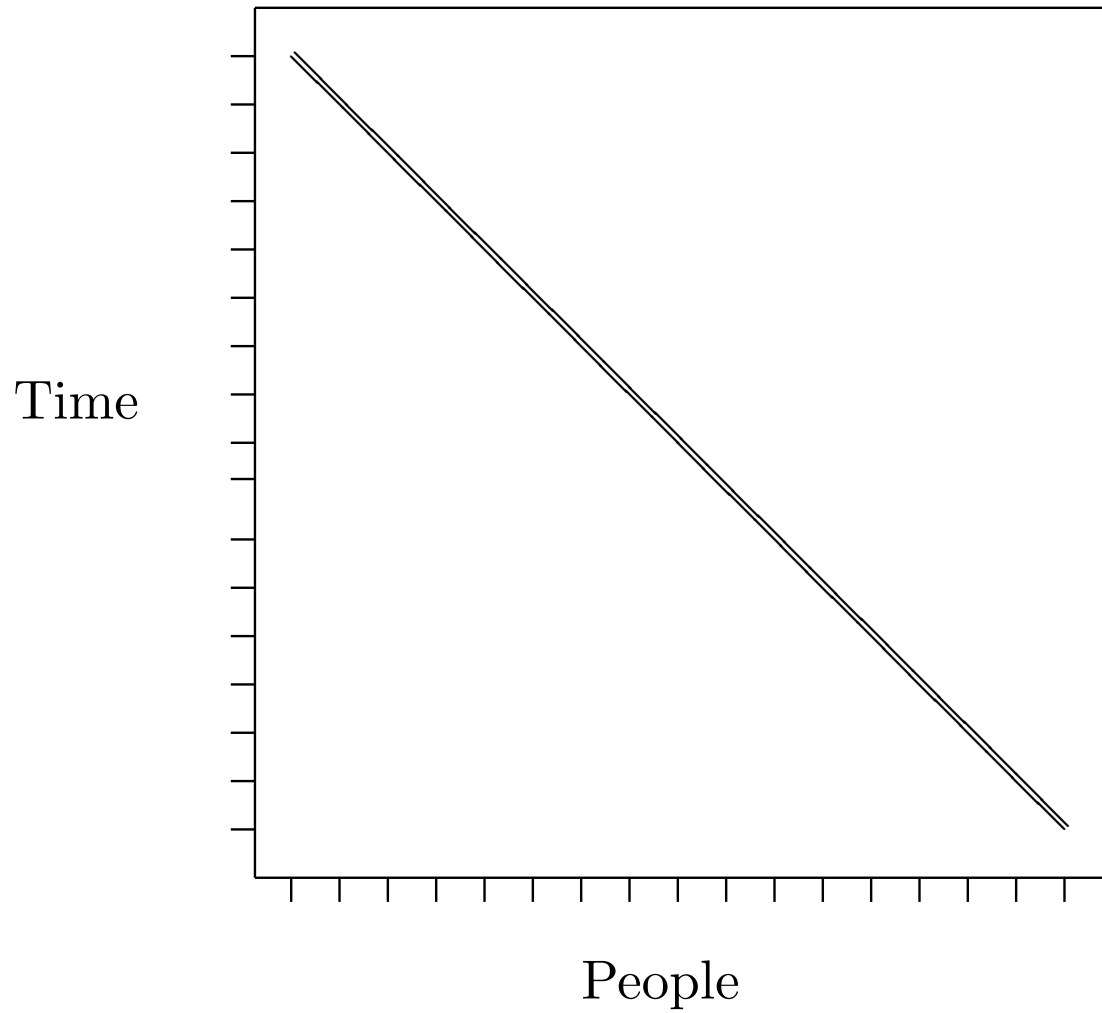
Question

1. What is the largest program that you have worked on?
2. For this program:
 - Approx. months/years to develop
 - Approx. lines of code (LOCs)
 - Approx. number of programmers
3. Was this a **small**, **medium** or **large** program?

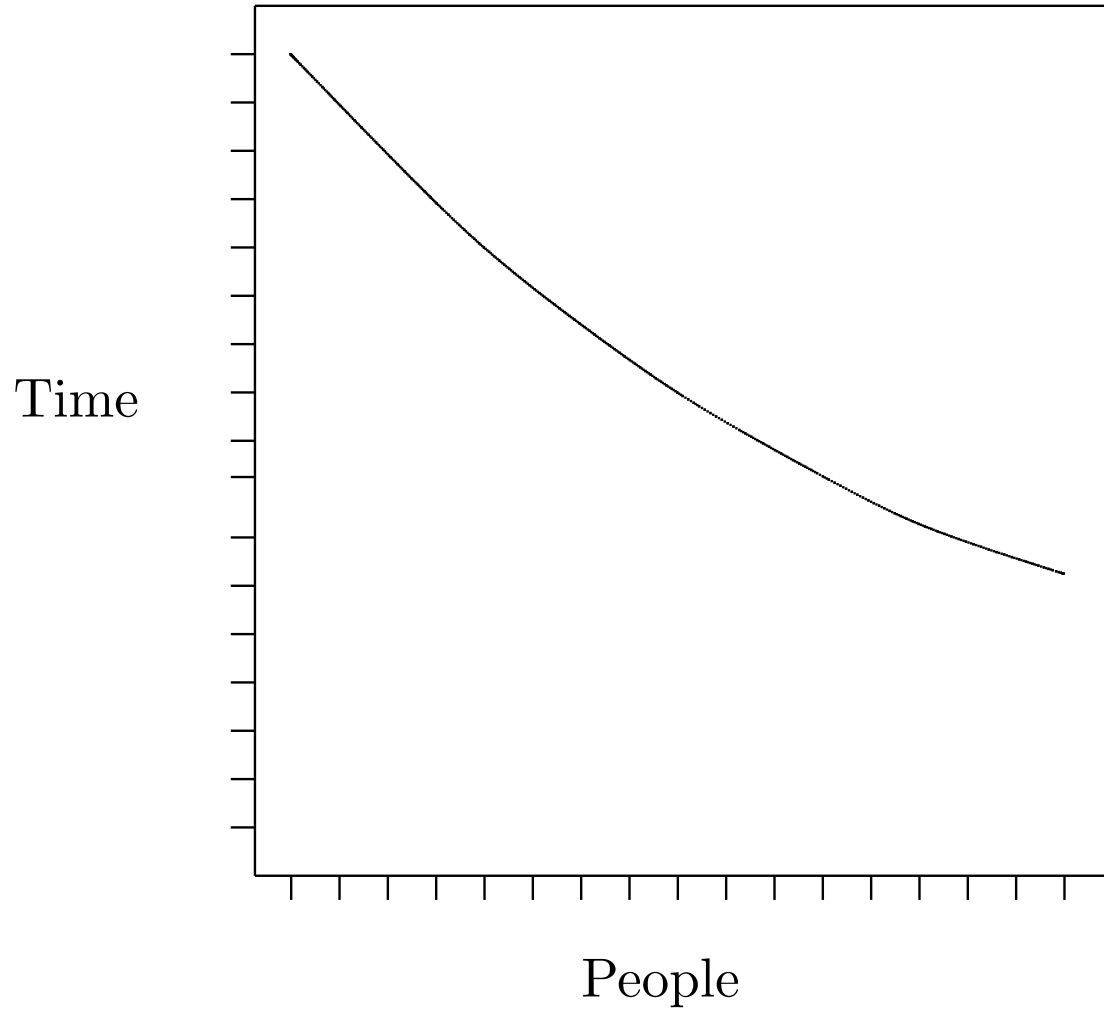
Mythical Man Month

- Why do we need to learn about software design?
- “More software projects have gone awry for lack of calendar time than for all other causes combined.”
- Brooks' Law: *Adding manpower to a late software project makes it later.*

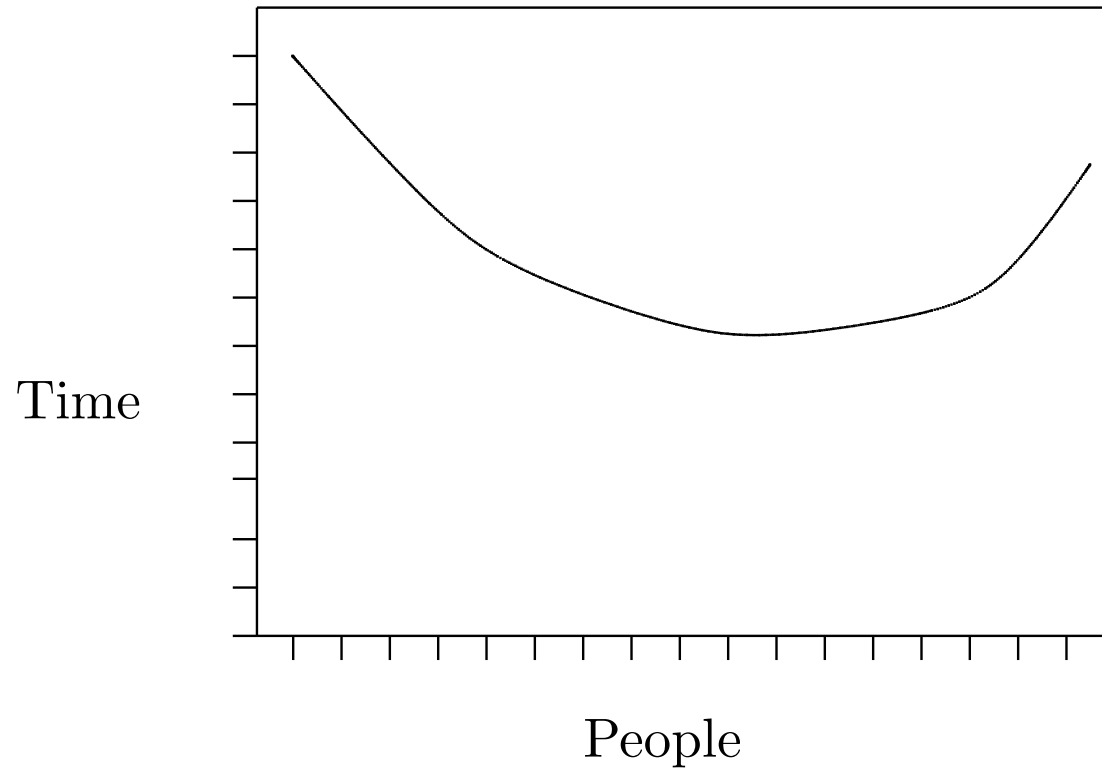
Ideal



Realistic



Depressingly Realistic



- Why is software problematic?
- What can we do about it?

Why is software problematic?

- Part of the problem is just poor planning and lack of discipline
- Why are programmers “bad” engineers?

Why is software problematic?

- Part of the problem is just poor planning and lack of discipline
- Why are programmers “bad” engineers?
- Software is intangible

Why is software problematic?

- Part of the problem is just poor planning and lack of discipline
- Why are programmers “bad” engineers?
- Software is intangible
 - Easy to reproduce
 - Easy to modify
- One person can do a lot

Why is software problematic?

- Part of the problem is just poor planning and lack of discipline
- Why are programmers “bad” engineers?
- Software is intangible
 - Easy to reproduce
 - Easy to modify
- One person can do a lot

Software is Flexible

- Consider a program that displays images
 - gif, jpeg, png, pbm, tiff, etc.

Software is Flexible

- Consider a program that displays images
 - gif, jpeg, png, pbm, tiff, etc.
 - Without too much trouble we could add support for video and audio

Software is Flexible

- Consider a program that displays images
 - gif, jpeg, png, pbm, tiff, etc.
 - Without too much trouble we could add support for video and audio
 - This does not seem too unreasonable.

Software is Flexible

- Consider a program that displays images
 - gif, jpeg, png, pbm, tiff, etc.
 - Without too much trouble we could add support for video and audio
 - This does not seem too unreasonable.
 - Consider a machine that :

Software is Flexible

- Consider a program that displays images
 - gif, jpeg, png, pbm, tiff, etc.
 - Without too much trouble we could add support for video and audio
 - This does not seem too unreasonable.
- Consider a machine that :
 - plays DVDS, CDs and VHS

Software is Flexible

- Consider a program that displays images
 - gif, jpeg, png, pbm, tiff, etc.
 - Without too much trouble we could add support for video and audio
 - This does not seem too unreasonable.
- Consider a machine that :
 - plays DVDS, CDs and VHS
 - What about BETA, Mini-disks, laser discs?

Software is Flexible

- Consider a program that displays images
 - gif, jpeg, png, pbm, tiff, etc.
 - Without too much trouble we could add support for video and audio
 - This does not seem too unreasonable.
- Consider a machine that :
 - plays DVDS, CDs and VHS
 - What about BETA, Mini-disks, laser discs?
 - What about Cassette tapes, 8 Tracks, LPs?

Flexibility: Blessing and a Curse

- Software is or can be
 - amazingly complex
 - easy to reproduce
 - easy to modify “physically”
 - not subject to aging, etc.
- All of these are good things, but can encourage bad behavior
- As software becomes more important, bad behavior becomes intolerable less tolerated

“No Silver Bullet”

- Bad programs can be written in any language
- Patterns can be misused
- Discipline and work will always be necessary