

CS194

4/7/11

What are the "software engineering" people saying?

Software Engineering: Theory and Practice, 4th Ed.

Shari Lawrence Pfleeger, Joanne M. Atlee
Prentice Hall, 2010

What are the "software engineering" people saying?

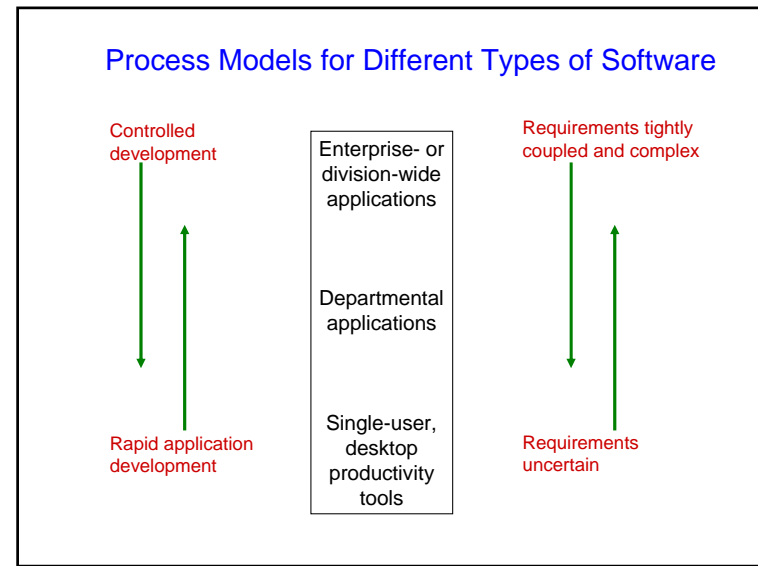
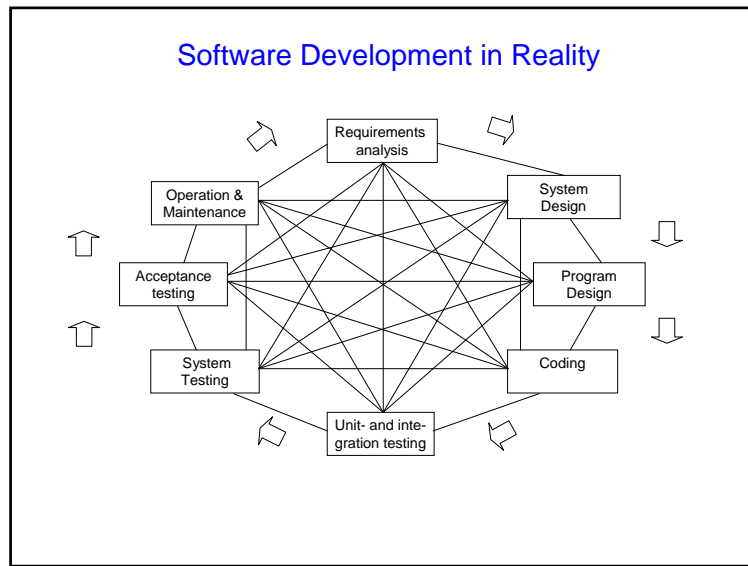
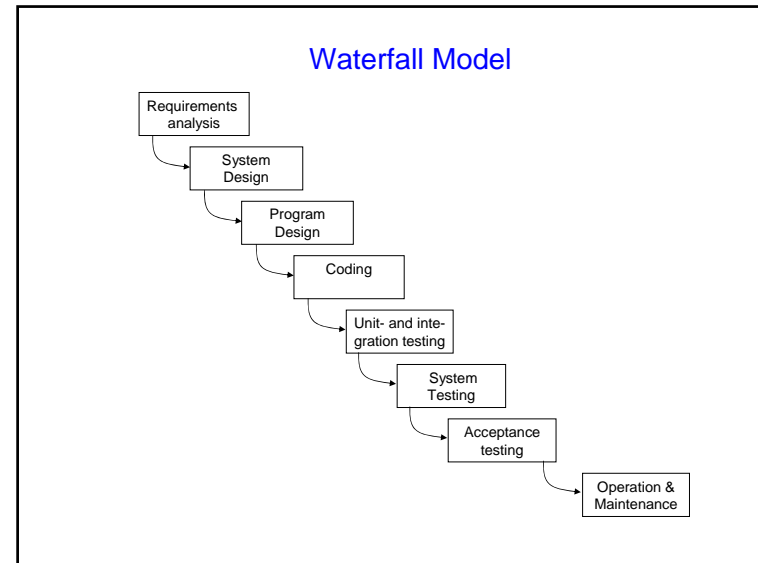
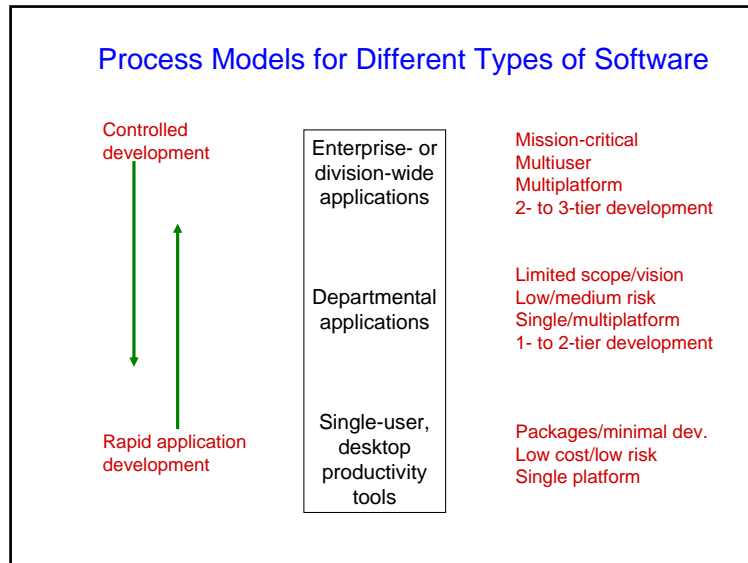
Software Engineering: Theory and Practice, 4th Ed.

Shari Lawrence Pfleeger, Joanne M. Atlee
Prentice Hall, 2010

Interesting statistic: software supporting the space shuttle has 3 million lines of code, with 100,000 lines of code in the shuttle itself (as of 1985).

Perspectives on Quality

Transcendental view	can recognize but not define
User view	fitness for purpose
Manufacturing view	conformance to specification
Product view	tied to internal product characteristics
Value-based view	amount the customer is willing to pay



**Remaining Chapters in
*Software Engineering: Theory and Practice***

- Planning and managing the project
- Capturing the requirements
- Designing the architecture
- Designing the modules
- Writing the programs
- Testing the programs
- Testing the system
- Delivering the system
- Maintaining the system
- Evaluating products, processes, and resources
- Improving predictions, products, processes, and resources
- The future of software engineering

**Chapters in
*Software Engineering: Theory and Practice***

- Why software engineering
- Modeling the process and life cycle
- Planning and managing the project
- Capturing the requirements
- Designing the architecture ←
- Designing the modules ← Last mentions of agile methods
- Writing the programs ←
- Testing the programs
- Testing the system
- Delivering the system
- Maintaining the system
- Evaluating products, processes, and resources
- Improving predictions, products, processes, and resources
- The future of software engineering

**No Silver Bullet--
Essence and Accident in
Software Engineering**

Frederick P. Brooks, Jr. (1986)

In:

The Mythical Man-Month
Essays on Software Engineering,
25th Anniversary Edition,
Addison-Wesley, 1995.

**No Silver Bullet--
Essence and Accident in
Software Engineering**

Frederick P. Brooks, Jr. (1986)

Could there be a silver bullet that would make software costs drop as rapidly as hardware costs do?

Hardware developments have been incredibly fast:

6 orders of magnitude in 30 years

We can take the gains either in improved performance
or reduced cost.

Hardware developments have been incredibly fast:

6 orders of magnitude in 30 years

We can take the gains either in improved performance
or reduced cost.

Could there be a technological or managerial technique that
would give even a single order of magnitude improvement
for software development?

Brooks divides software difficulties into two parts:

Essence -- the difficulties that are inherent
in the nature of software.

Accident -- the difficulties that are currently involved
in the production of software but that
are not inherent.

Brooks:

The essence of a software entity is a construct of interlocking concepts:
data sets, relationships among data items, algorithms, and
invocations of functions.

Brooks:

The essence of a software entity is a construct of interlocking concepts: data sets, relationships among data items, algorithms, and invocations of functions.

This essence is abstract, in that the conceptual construct is the same under many different representations.

Brooks:

The essence of a software entity is a construct of interlocking concepts: data sets, relationships among data items, algorithms, and invocations of functions.

This essence is abstract, in that the conceptual construct is the same under many different representations.

I believe the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation.

Brooks:

The essence of a software entity is a construct of interlocking concepts: data sets, relationships among data items, algorithms, and invocations of functions.

This essence is abstract, in that the conceptual construct is the same under many different representations.

I believe the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation.

If this is true, building software will always be hard.
There is inherently no silver bullet.

The inherent properties of software:

Complexity

No two parts are the same (unlike buildings, automobiles, etc.)

Scaling up increases the number of different elements, and complexity increases more than linearly due to interactions.

Difficulties due to complexity

Communication among team members

Enumerating (and understanding) the possible states of a program

Extending programs without unwanted side effects

Maintaining an overview of a project

Coping with staff turnover

The inherent properties of software:

Conformity

Complexity is forced upon software by the human institutions and systems to which it must conform.

We are called upon to "preserve the mess".

Changeability

There are constant pressures for change

All successful software gets changed

Use is extended beyond the original domain

The software outlives the hardware

The inherent properties of software:

Invisibility

Software is not inherently embedded in space. There are no geometric abstractions as powerful as floor plans, scale drawings, models, etc.

Diagramming software helps some, but introduces a complexity of its own.

Past breakthroughs solved accidental difficulties

High-level languages remove much of the accidental complexity

Time-sharing and personal computers

Integrated development environments

Hopes for the Silver

New languages

Object-oriented programming

AI/Expert systems/Automatic programming

Graphical programming

Program verification

Environments and tools

Promising attacks on conceptual essence

Buy versus build

Requirements refinement and rapid prototyping

Incremental development--grow, not build, software

Great designers

No Silver Bullet

To Do

Turn in your proposal as per my email

Sign up for writing tutorial if you need to

Send in your slides by Sunday, 11:59 pm

Specify any preference for Tues/Thurs

We will notify you of which day you will present

Sign-ups for weekly meetings will be next week