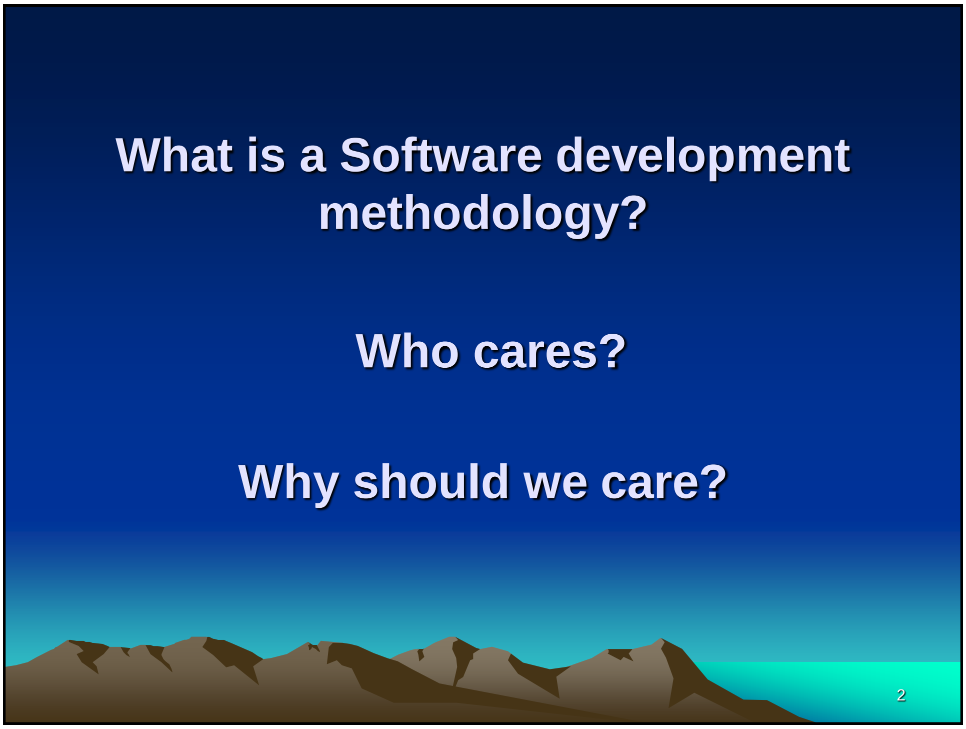
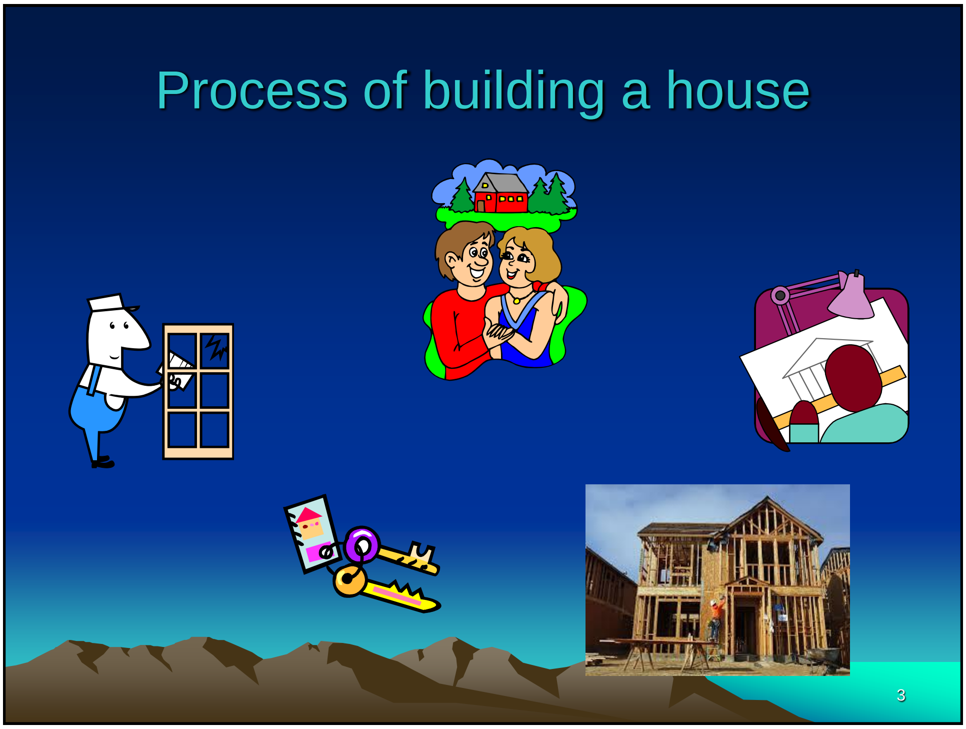
Introduction to Software Engineering



What is a Software development methodology?

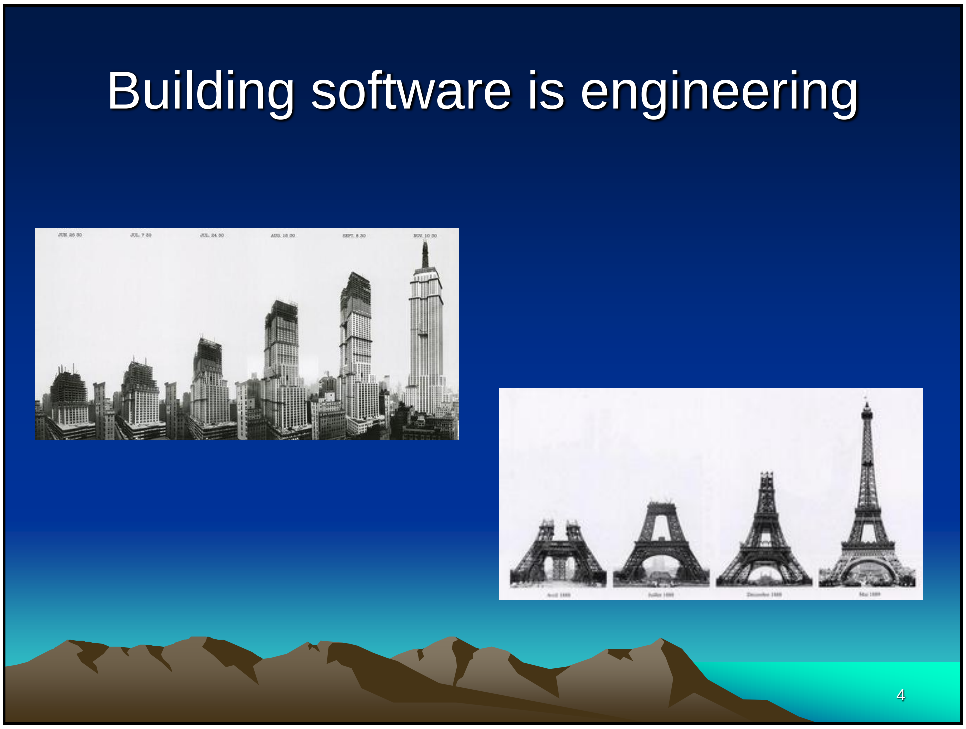
Who cares?

Why should we care?



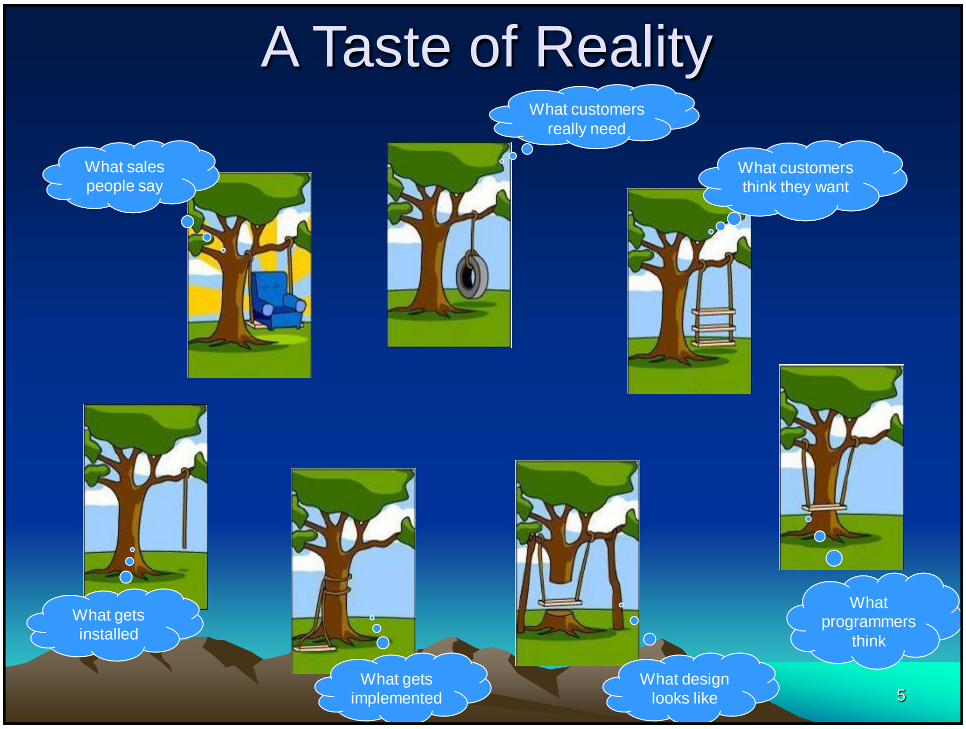
Process of building a house

3



Building software is engineering

4



A Taste of Reality

What sales people say

What customers really need

What customers think they want

What programmers think

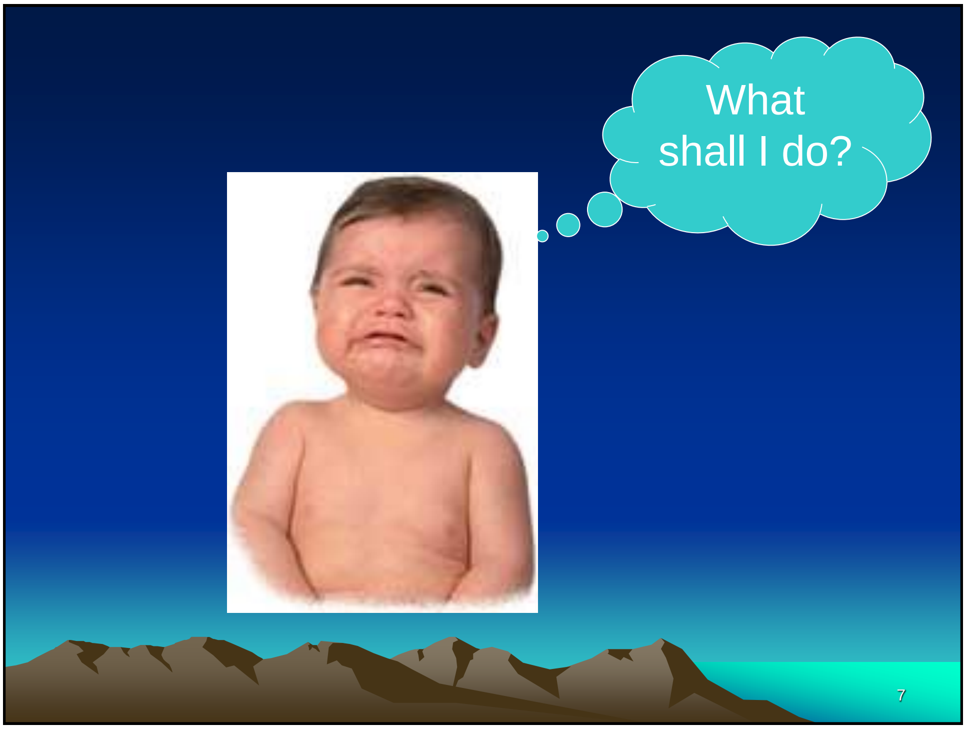
What gets installed

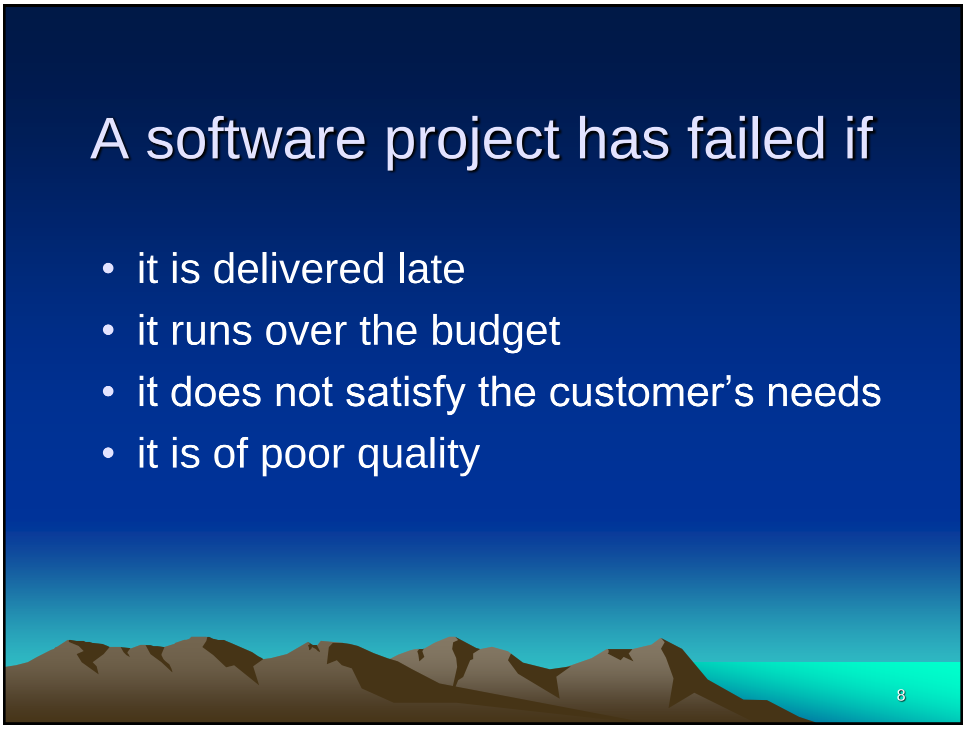
What gets implemented

What design looks like

Customers Care!

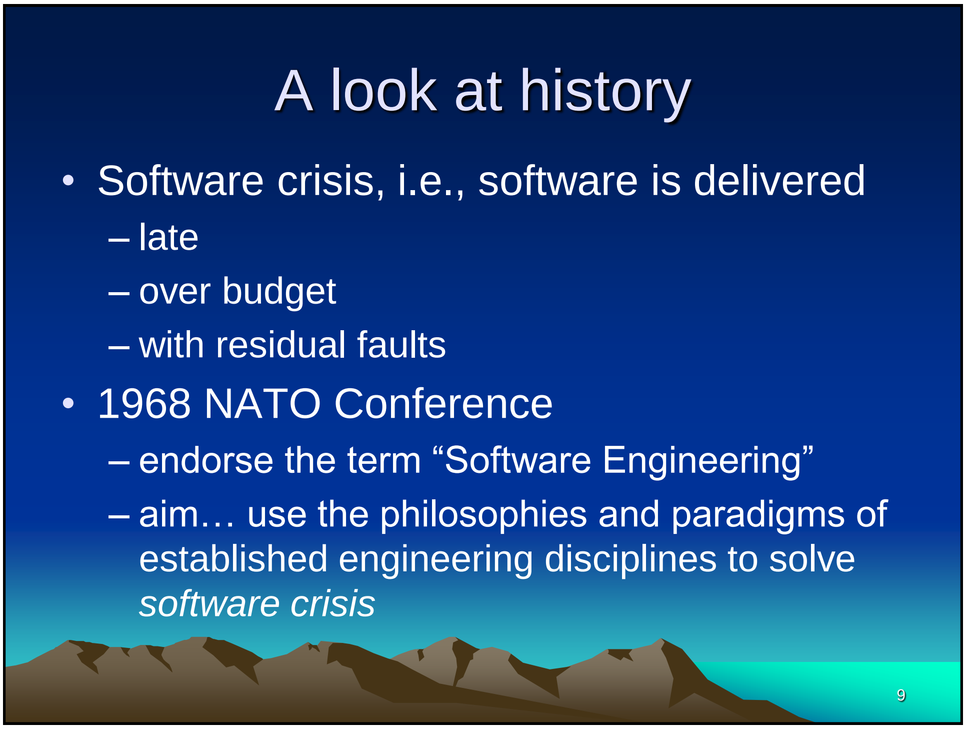
What customers gets charged





A software project has failed if

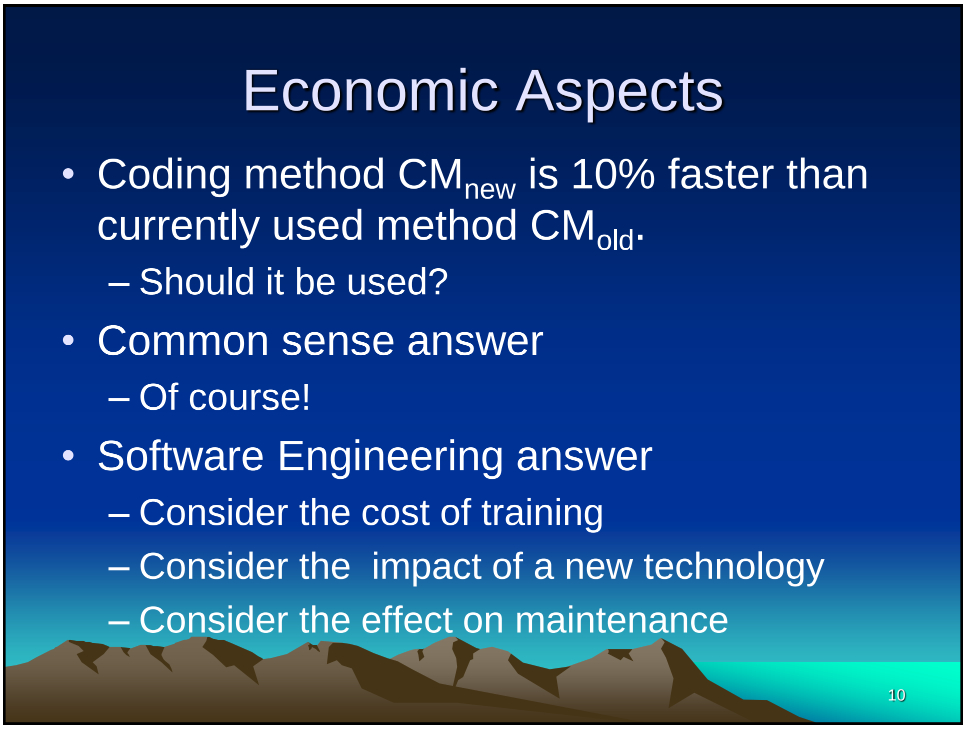
- it is delivered late
- it runs over the budget
- it does not satisfy the customer’s needs
- it is of poor quality



A look at history

- Software crisis, i.e., software is delivered
  - late
  - over budget
  - with residual faults
- 1968 NATO Conference
  - endorse the term “Software Engineering”
  - aim… use the philosophies and paradigms of established engineering disciplines to solve *software crisis*

9

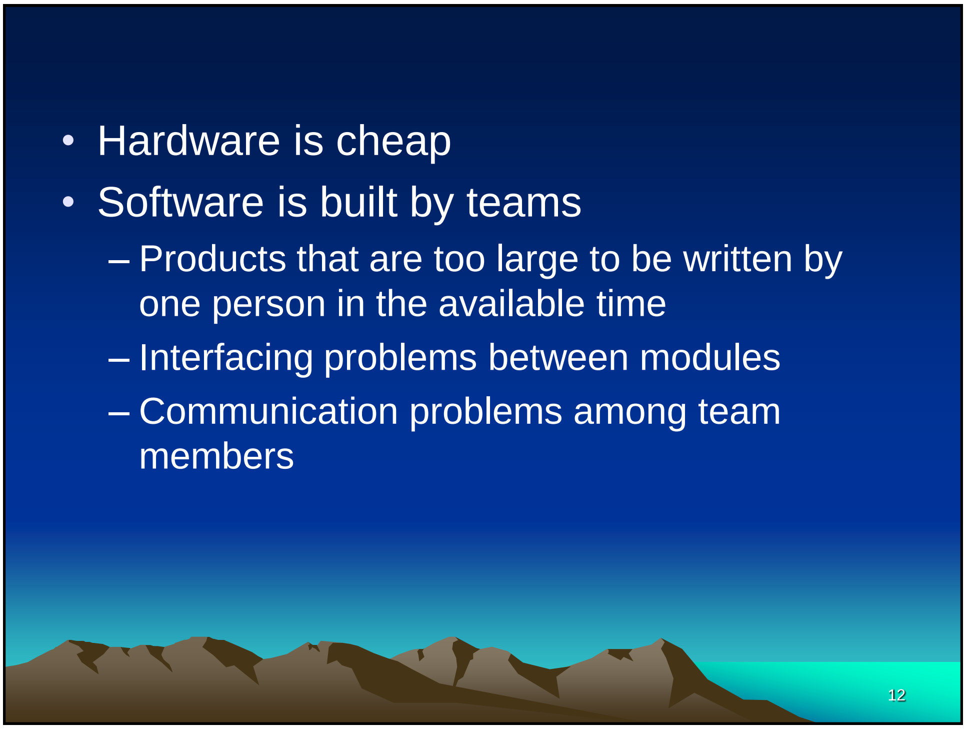


Economic Aspects

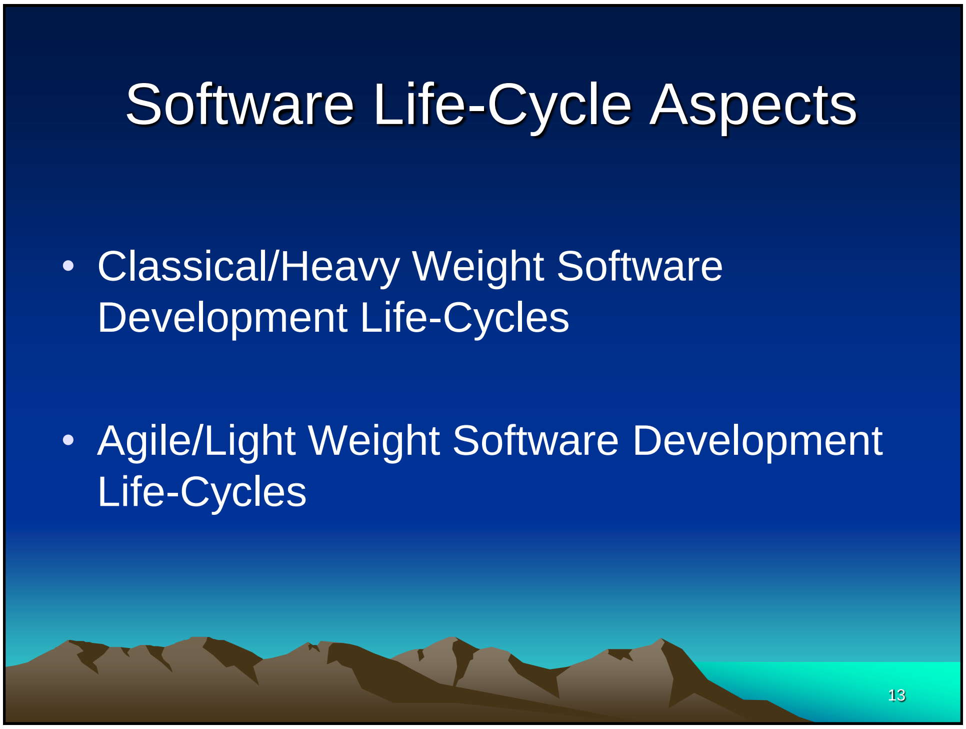
- Coding method CM_{new} is 10% faster than currently used method CM_{old}.
  - Should it be used?
- Common sense answer
  - Of course!
- Software Engineering answer
  - Consider the cost of training
  - Consider the impact of a new technology
  - Consider the effect on maintenance

10

Team Programming Aspects



- Hardware is cheap
- Software is built by teams
  - – Products that are too large to be written by one person in the available time
  - – Interfacing problems between modules
  - – Communication problems among team members



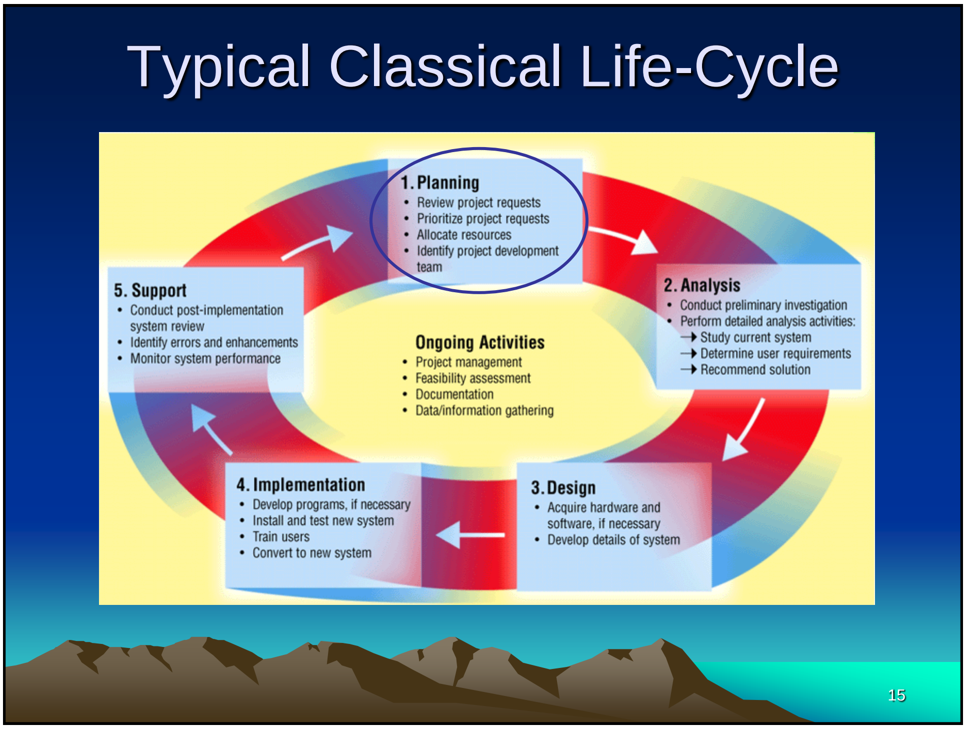
Software Life-Cycle Aspects

- Classical/Heavy Weight Software Development Life-Cycles
- Agile/Light Weight Software Development Life-Cycles

13

Classical Software Development Methodologies

14



Typical Classical Life-Cycle

1. Planning
- Review project requests
- Prioritize project requests
- Allocate resources
- Identify project development team

2. Analysis
- Conduct preliminary investigation
- Perform detailed analysis activities:
  - → Study current system
  - → Determine user requirements
  - → Recommend solution

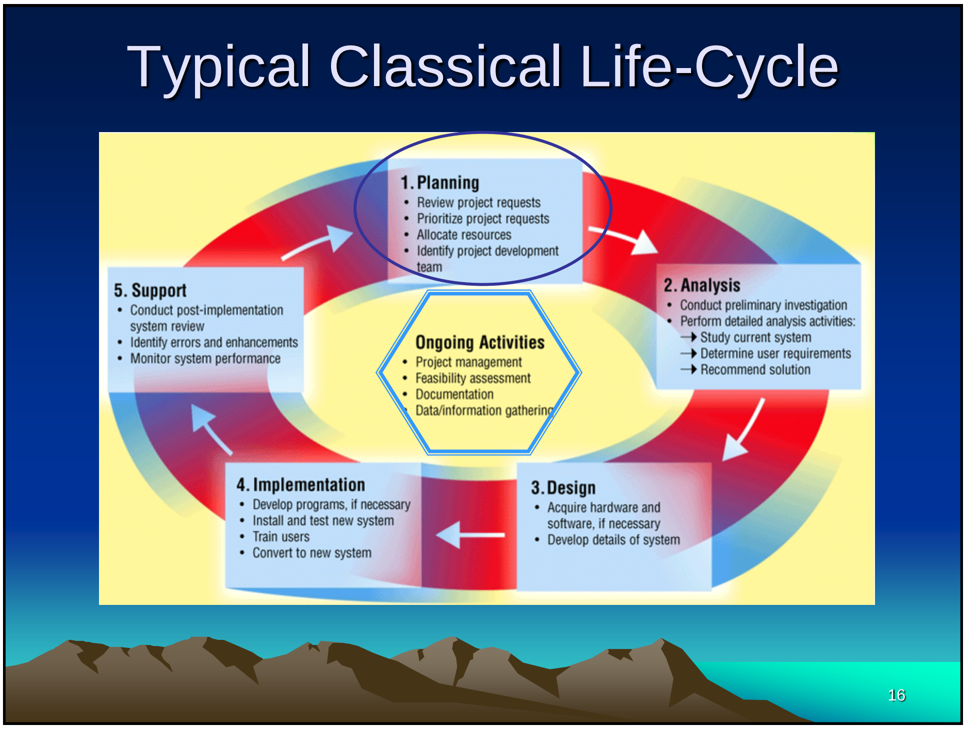
3. Design
- Acquire hardware and software, if necessary
- Develop details of system

4. Implementation
- Develop programs, if necessary
- Install and test new system
- Train users
- Convert to new system

5. Support
- Conduct post-implementation system review
- Identify errors and enhancements
- Monitor system performance

Ongoing Activities
- Project management
- Feasibility assessment
- Documentation
- Data/information gathering

15



Typical Classical Life-Cycle

1. Planning
- Review project requests
- Prioritize project requests
- Allocate resources
- Identify project development team

2. Analysis
- Conduct preliminary investigation
- Perform detailed analysis activities:
  - → Study current system
  - → Determine user requirements
  - → Recommend solution

3. Design
- Acquire hardware and software, if necessary
- Develop details of system

4. Implementation
- Develop programs, if necessary
- Install and test new system
- Train users
- Convert to new system

5. Support
- Conduct post-implementation system review
- Identify errors and enhancements
- Monitor system performance

Ongoing Activities
- Project management
- Feasibility assessment
- Documentation
- Data/information gathering

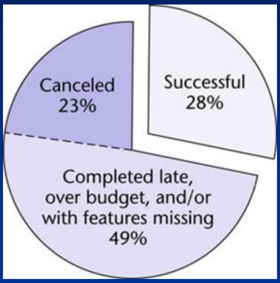
16

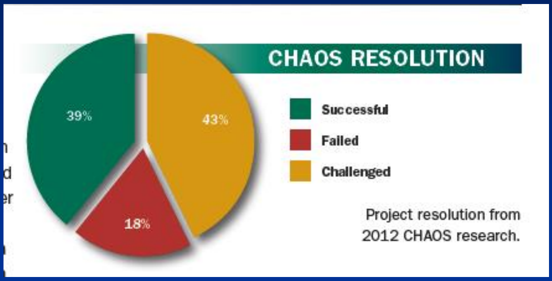
Afghanistan School Of Suicide Bombers
Now pay attention, I'm only going to do this once!

17

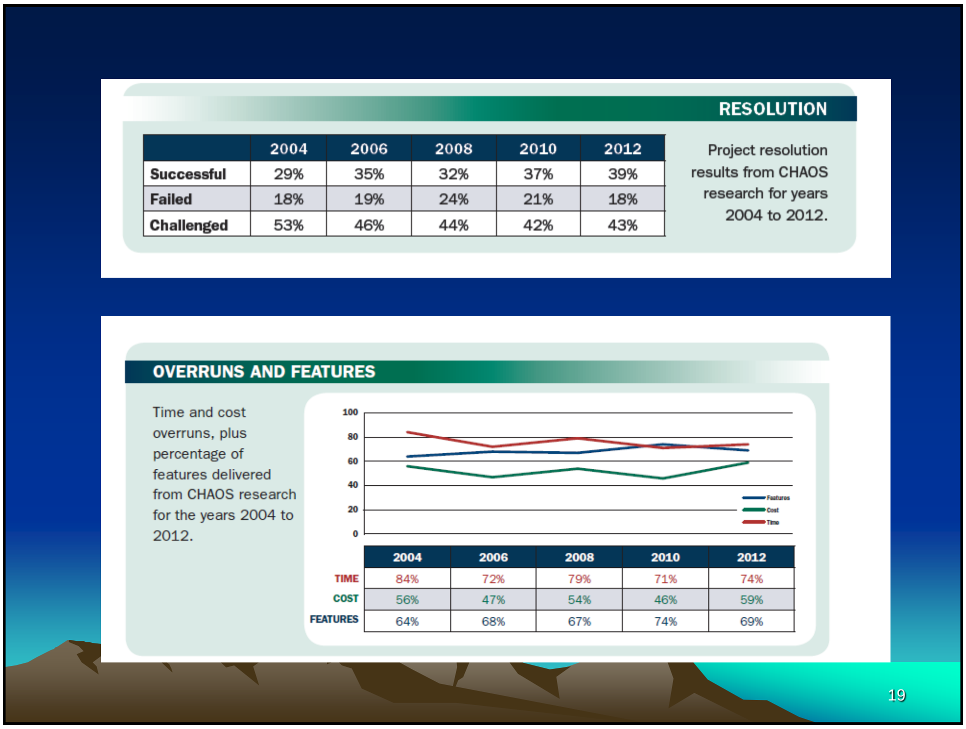
Standish group data

(Data on 28,000 projects completed in 2000)





18



RESOLUTION

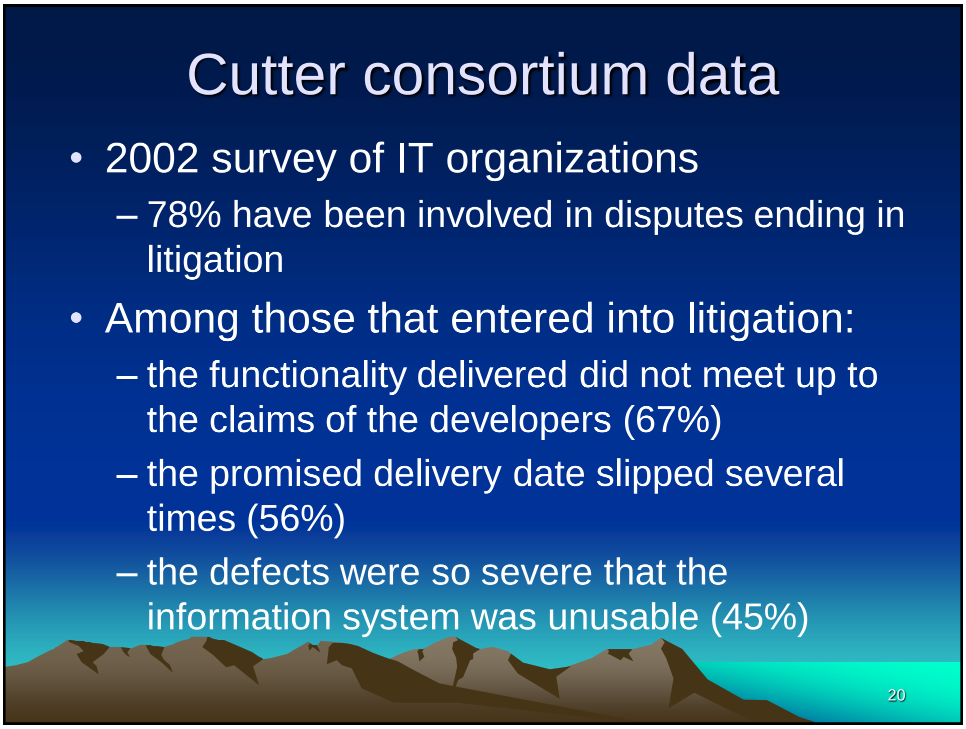
	2004	2006	2008	2010	2012
Successful	29%	35%	32%	37%	39%
Failed	18%	19%	24%	21%	18%
Challenged	53%	46%	44%	42%	43%

Project resolution results from CHAOS research for years 2004 to 2012.

OVERRUNS AND FEATURES

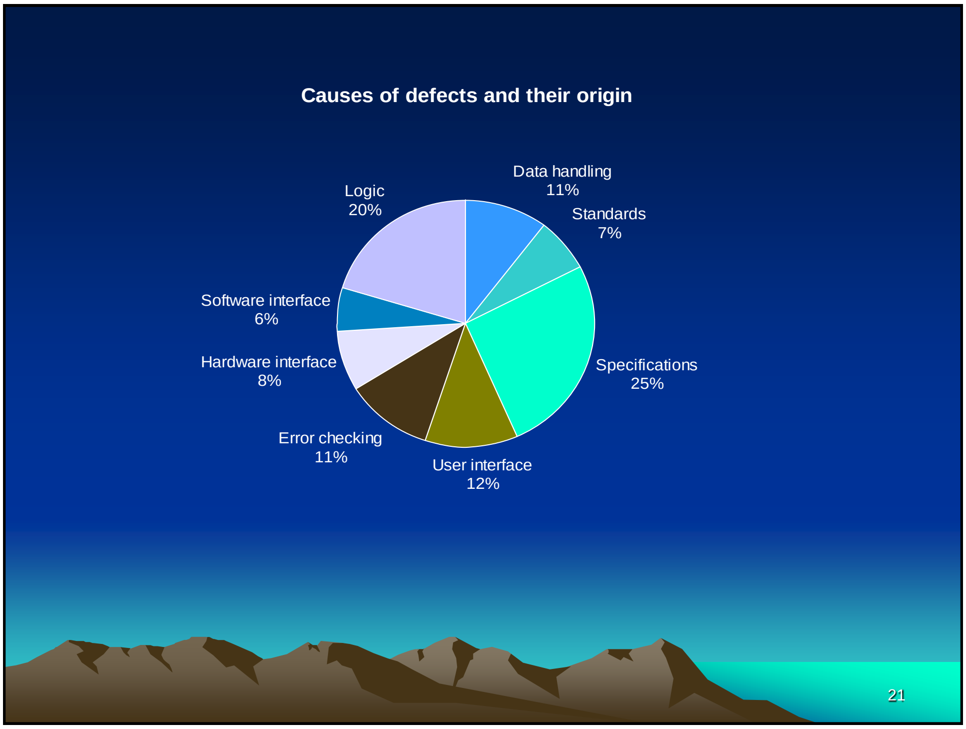
Time and cost overruns, plus percentage of features delivered from CHAOS research for the years 2004 to 2012.

	2004	2006	2008	2010	2012
TIME	84%	72%	79%	71%	74%
COST	56%	47%	54%	46%	59%
FEATURES	64%	68%	67%	74%	69%

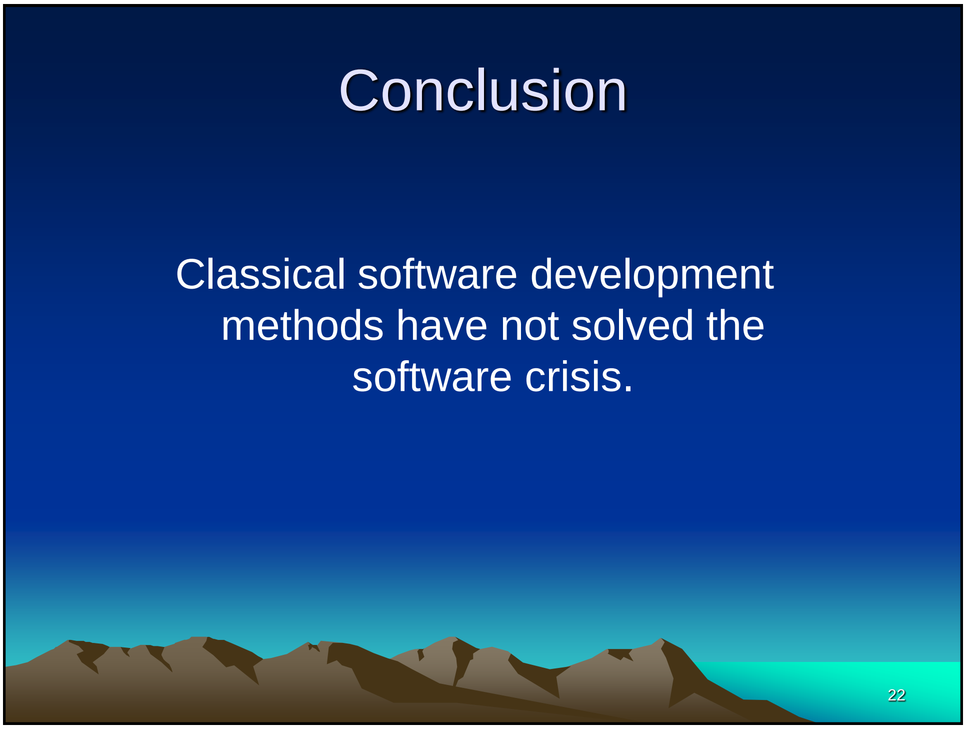


Cutter consortium data

- 2002 survey of IT organizations
  - 78% have been involved in disputes ending in litigation
- Among those that entered into litigation:
  - the functionality delivered did not meet up to the claims of the developers (67%)
  - the promised delivery date slipped several times (56%)
  - the defects were so severe that the information system was unusable (45%)



Causes of defects and their origin



Conclusion

Classical software development methods have not solved the software crisis.

Agile Software Development Methodology

To satisfy the customer through early and continuous delivery of valuable software

Agile Software Development Methods - Review and Analysis

23

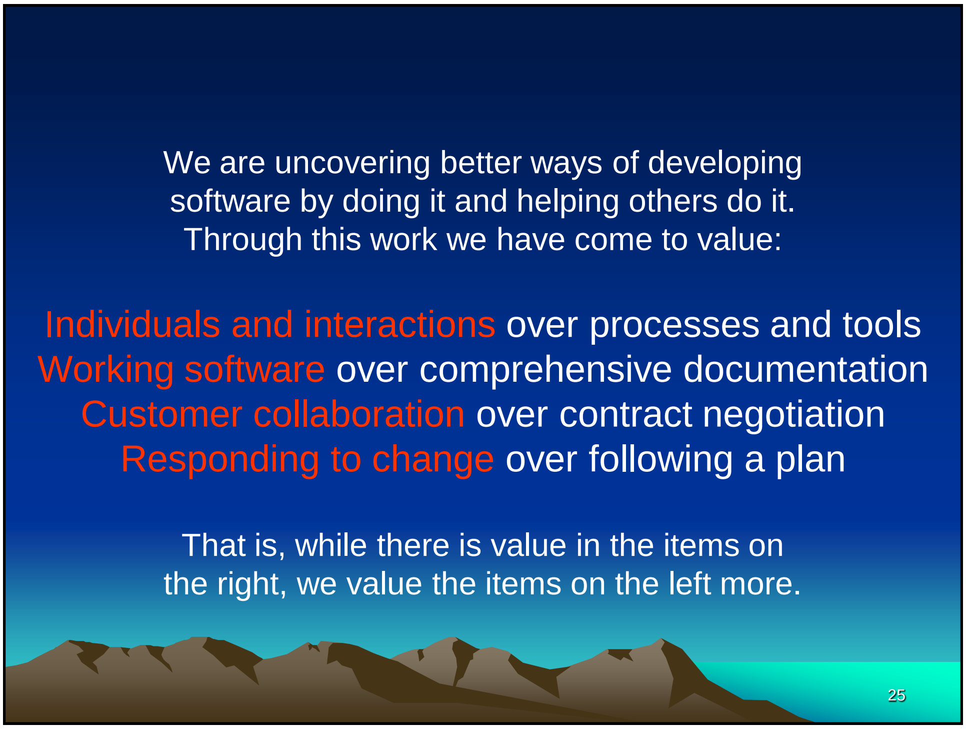
Manifesto for Agile Software Development

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham

Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern

Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

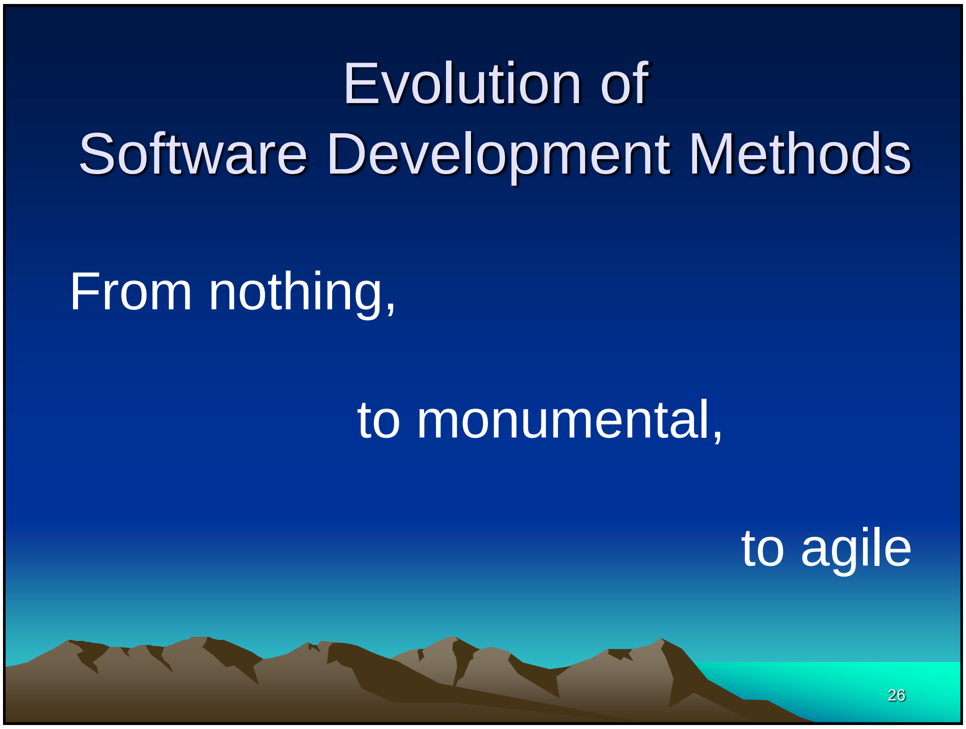
24



We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

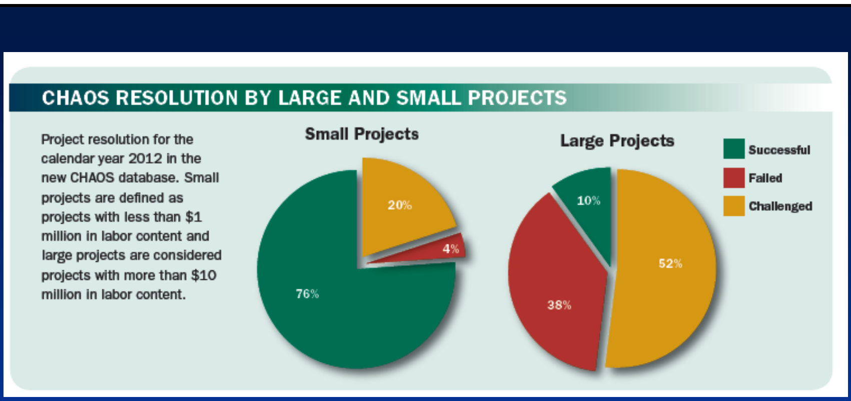


Evolution of Software Development Methods

From nothing,

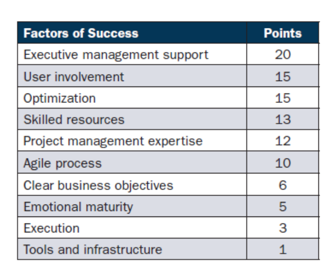
to monumental,

to agile



CHAOS RESOLUTION BY LARGE AND SMALL PROJECTS

Project resolution for the calendar year 2012 in the new CHAOS database. Small projects are defined as projects with less than $1 million in labor content and large projects are considered projects with more than $10 million in labor content.



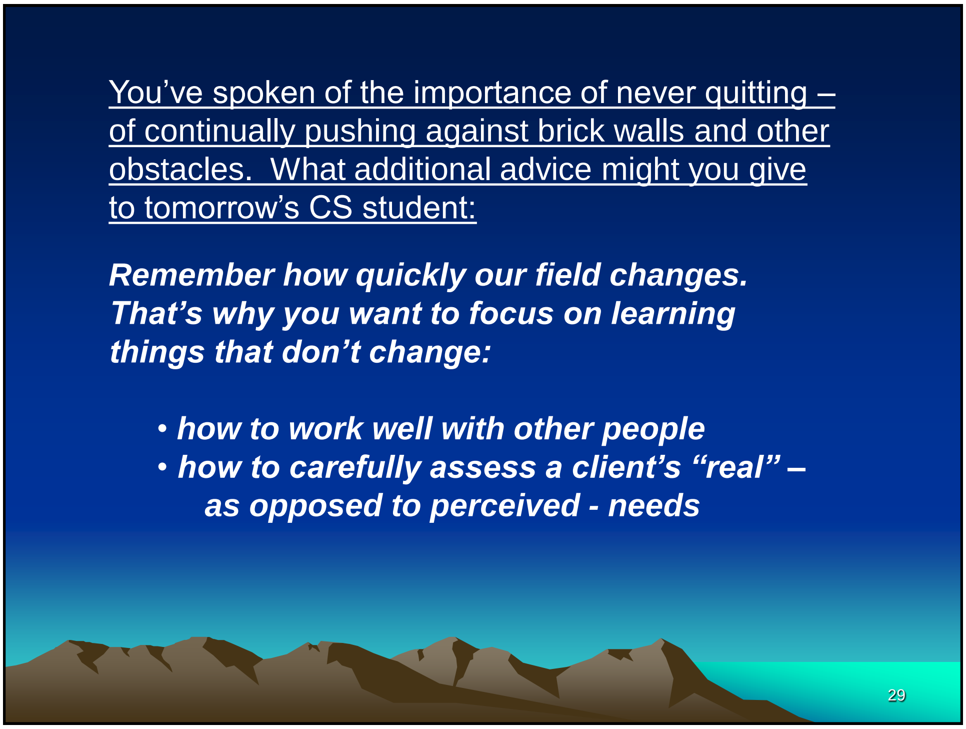
Factors of Success	Points
Executive management support	20
User involvement	15
Optimization	15
Skilled resources	13
Project management expertise	12
Agile process	10
Clear business objectives	6
Emotional maturity	5
Execution	3
Tools and infrastructure	1

Last Lecture given by a CS Professor at Carnegie Mellon University about Achieving Your Childhood Dream after his doctor told him that he could only live for 6 months

3D Programming Language Alice

Virtual Reality

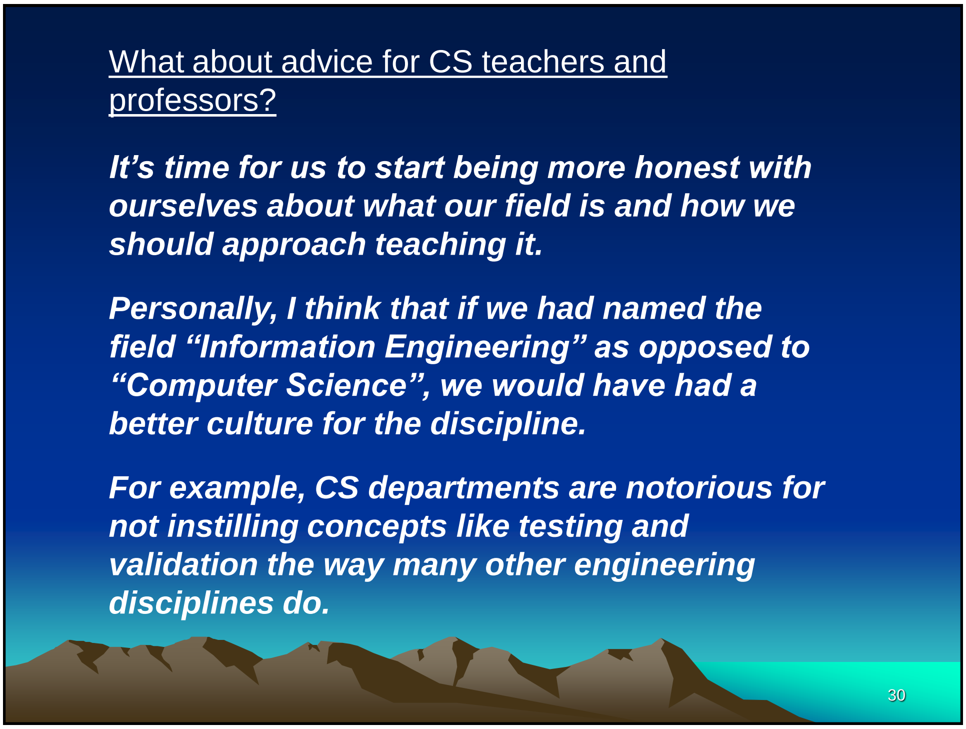
Randy Pausch



You've spoken of the importance of never quitting – of continually pushing against brick walls and other obstacles. What additional advice might you give to tomorrow's CS student:

Remember how quickly our field changes. That's why you want to focus on learning things that don't change:

- ***how to work well with other people***
- ***how to carefully assess a client's "real" – as opposed to perceived - needs***

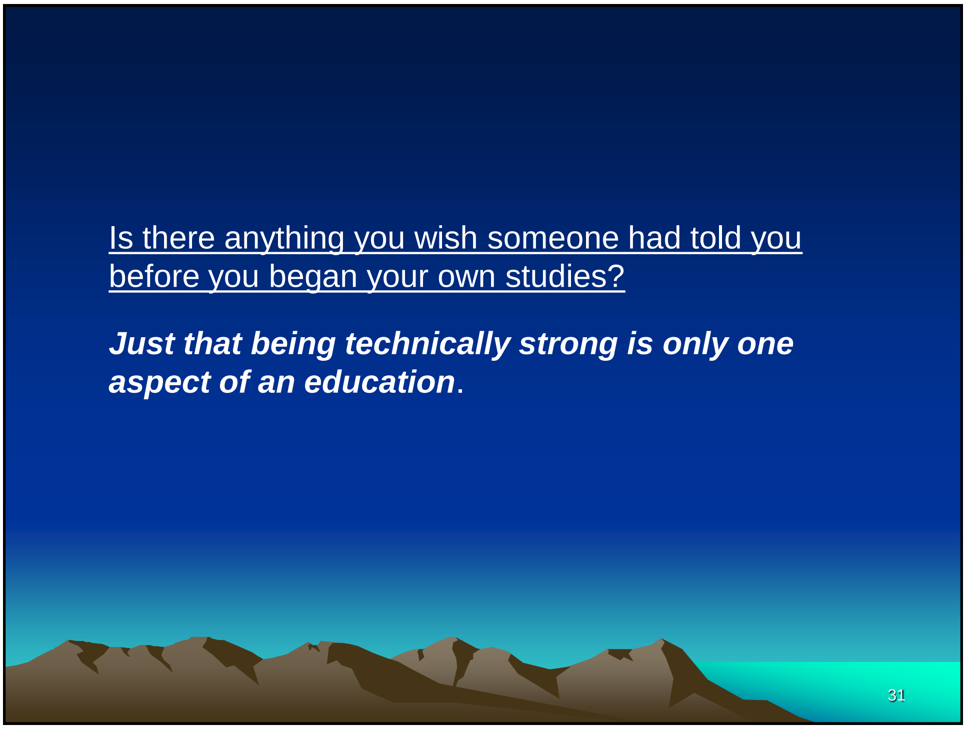


What about advice for CS teachers and professors?

It's time for us to start being more honest with ourselves about what our field is and how we should approach teaching it.

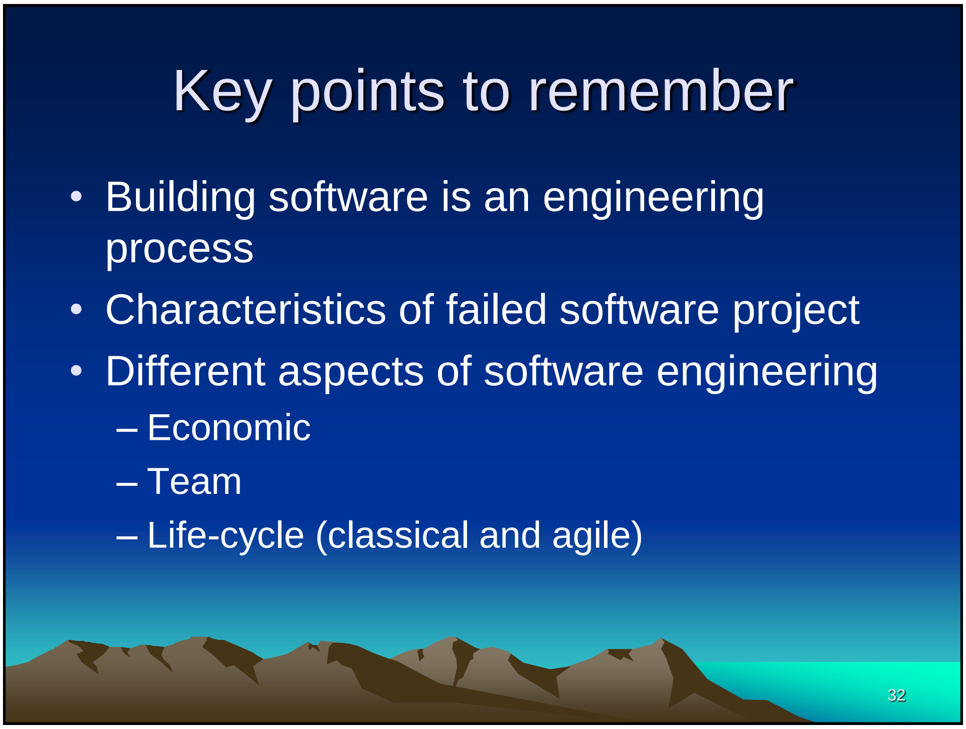
Personally, I think that if we had named the field "Information Engineering" as opposed to "Computer Science", we would have had a better culture for the discipline.

For example, CS departments are notorious for not instilling concepts like testing and validation the way many other engineering disciplines do.



Is there anything you wish someone had told you before you began your own studies?

***Just that being technically strong is only one aspect of an education*.**



Key points to remember

- Building software is an engineering process
- Characteristics of failed software project
- Different aspects of software engineering
  - Economic
  - Team
  - Life-cycle (classical and agile)