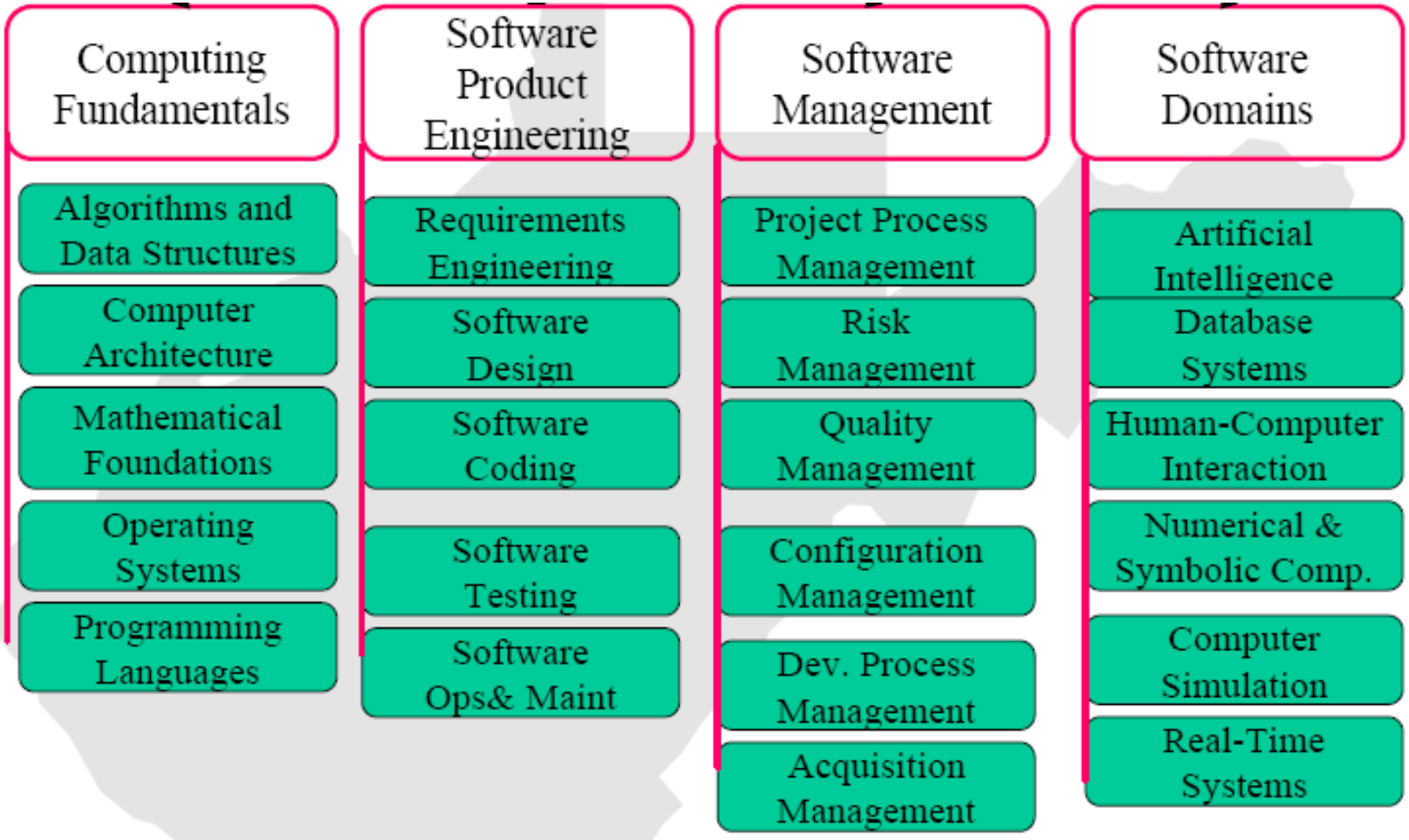
What is the difference between software engineering and computer science?

Computer Science	Software Engineering
is concerned with	
theory fundamentals Algorithms, date structures, complexity theory, numerical methods	Understanding domain challenges the practicalities of developing and delivering useful quality software SE deals with practical problems in complex software products

Computer science theories are currently *insufficient* to act as a complete underpinning for software engineering, BUT they provide a foundation for practical aspects of software engineering

Software Engineering Body of Knowledge



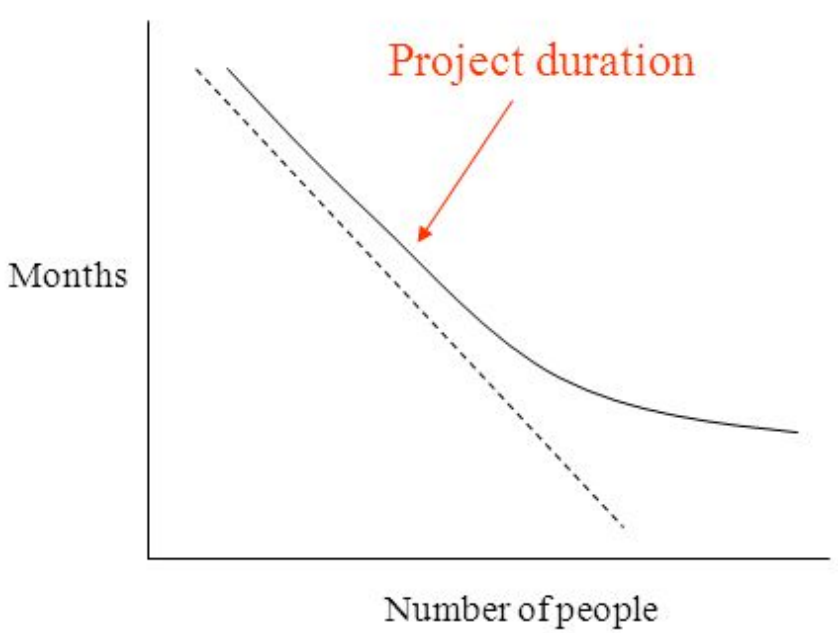
Source: http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99tr004.pdf

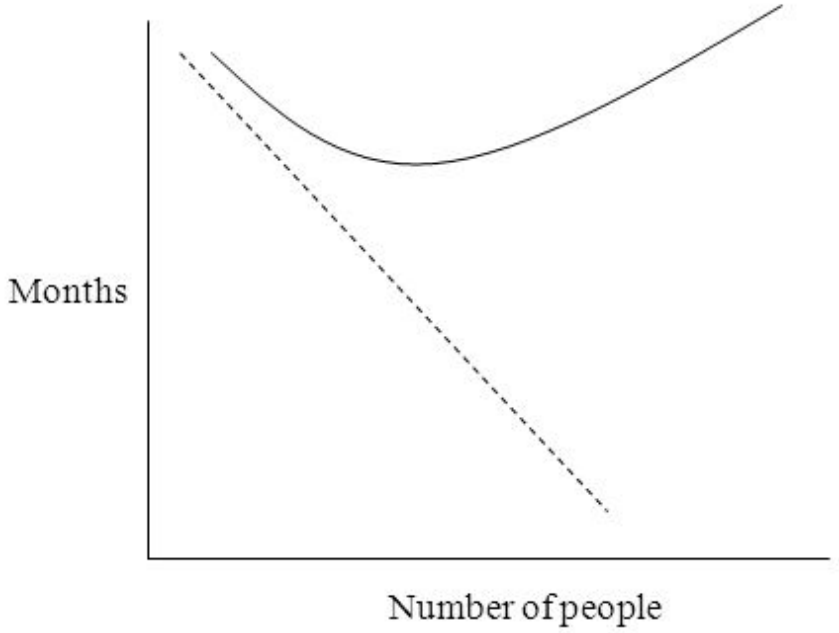
SE history

- SE introduced first in 1968
  - conference about "software crisis"- when the introduction of third generation computer hardware led to developing more complex software systems than before
- Early approaches, based on informal methodologies, led to
  - **Delays** in software delivery
  - **Higher costs** than initially estimated
  - **Unreliable**, difficult to **maintain** software

- Thus, there is a need for new methods and techniques to manage the production of complex software, ones that consider the *intangible* nature of software as a product.

Does adding more people resolve the delay in software delivery?





Due to sequential constraints
the relationship will not be linear

Fred Brook, Hypothetical Man month,

Software myths

- **Management myths**
  - *Standards and procedures for building software exist*
  - *Add more programmers if behind schedule*
- **Customer myths**
  - *A general description of objectives enough to start coding*
  - *Requirements may change as software is flexible*
- **Practitioner myths**
  - *Task accomplished when the program works*
  - *Quality assessment when the program is running*
  - *"Working program" the only project deliverable*

Why Software Engineering?

Why do we need Software Engineering?

Software failures

Software failures

- **Therac-25 (1985-1987)**: six people overexposed during treatments for cancer
- **Taurus (1993)**: the planned automatic transaction settlement system for London Stock Exchange cancelled after five years of development
- **Ariane 5 (1996)**: rocket exploded soon after its launch due error conversion (16 floating point into 16-bit integer)
- **The Mars Climate Orbiter** assumed to be lost by NASA officials (1999): different measurement systems (Imperial and metric)

More Software failures

- **Passport System** delays cause backlog (1999, UK)
- **Ferry Company** left thousands of lorries stranded for 12 hours (back up also failed, 1999, UK)
- **Inland Revenue** (IR) 'losing tax records' (2000, UK)
  => IR spokesman said 'All major IT initiatives have some kind of teething problems ....'
  => Guardian (20 July 2000) 'At the centre of the crisis are two computer systems .... Files appear to have gone missing somewhere between the two'
- **General Motors Ford** Cars (2016, USA + Worldwide): A "software bug" that may cause human safety, 4.5M cars recalled.

Even More Software failures

In 1995 annual US spending on software projects reached 250 billion dollars

This involved some 175,000 projects

Of this spend:

- Overspend cost 59 billion dollars
- Cancelled projects cost 81 billion dollars

Software Failures

Why does a software system fail?

Causes of software failure

Causes of Software Failure

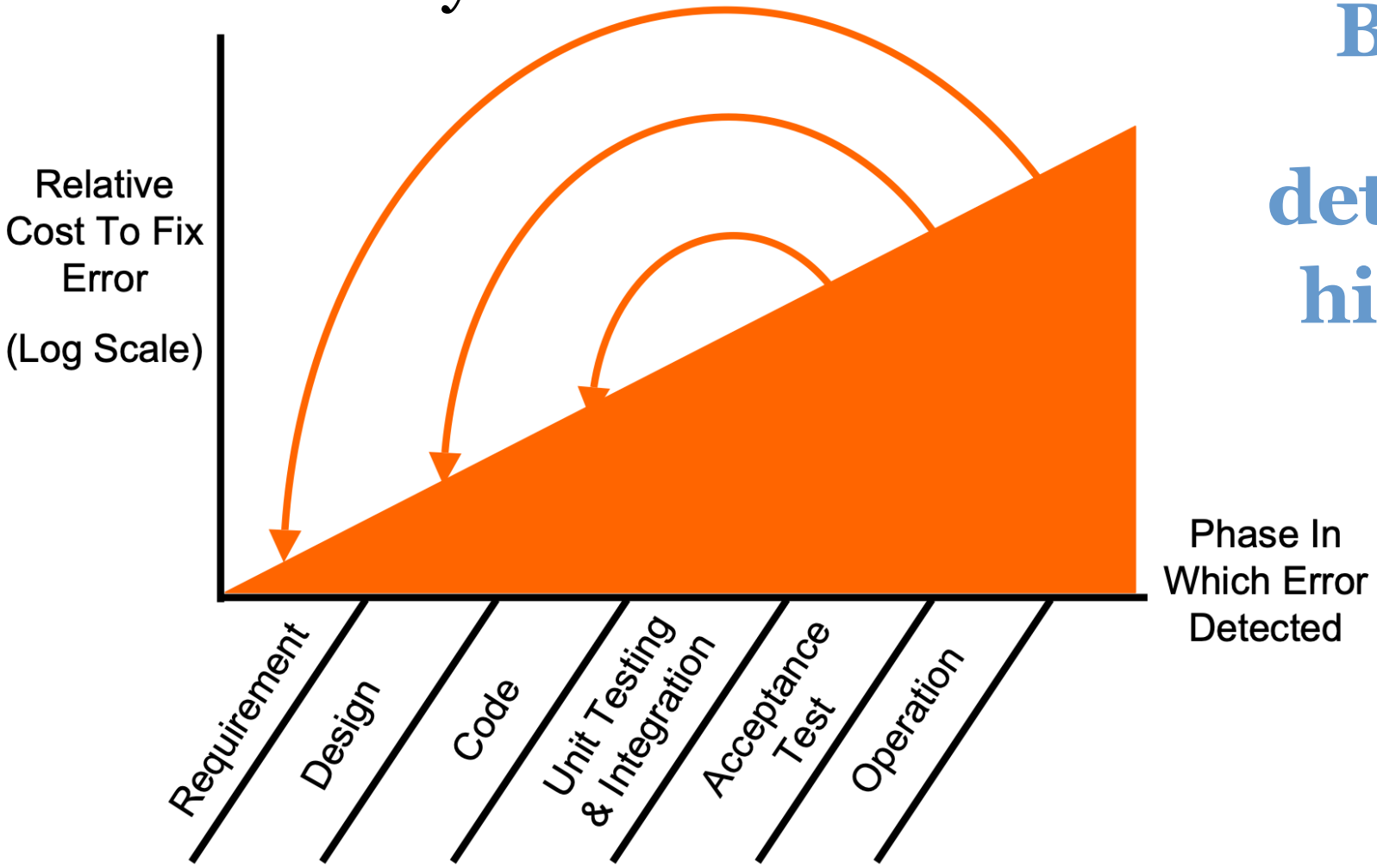
Many factors can cause software failure, however, there are some general causes, including:

-Undetected bugs!
-Co-evolution of software
-Costs factors
-Risk factors

Greater complexity= greater changes = potential errors!

Causes: Bugs

Cost of delayed error detection



Bugs: the later detected, the higher cost to fix

Causes: IT System co-evolution- eternal loop