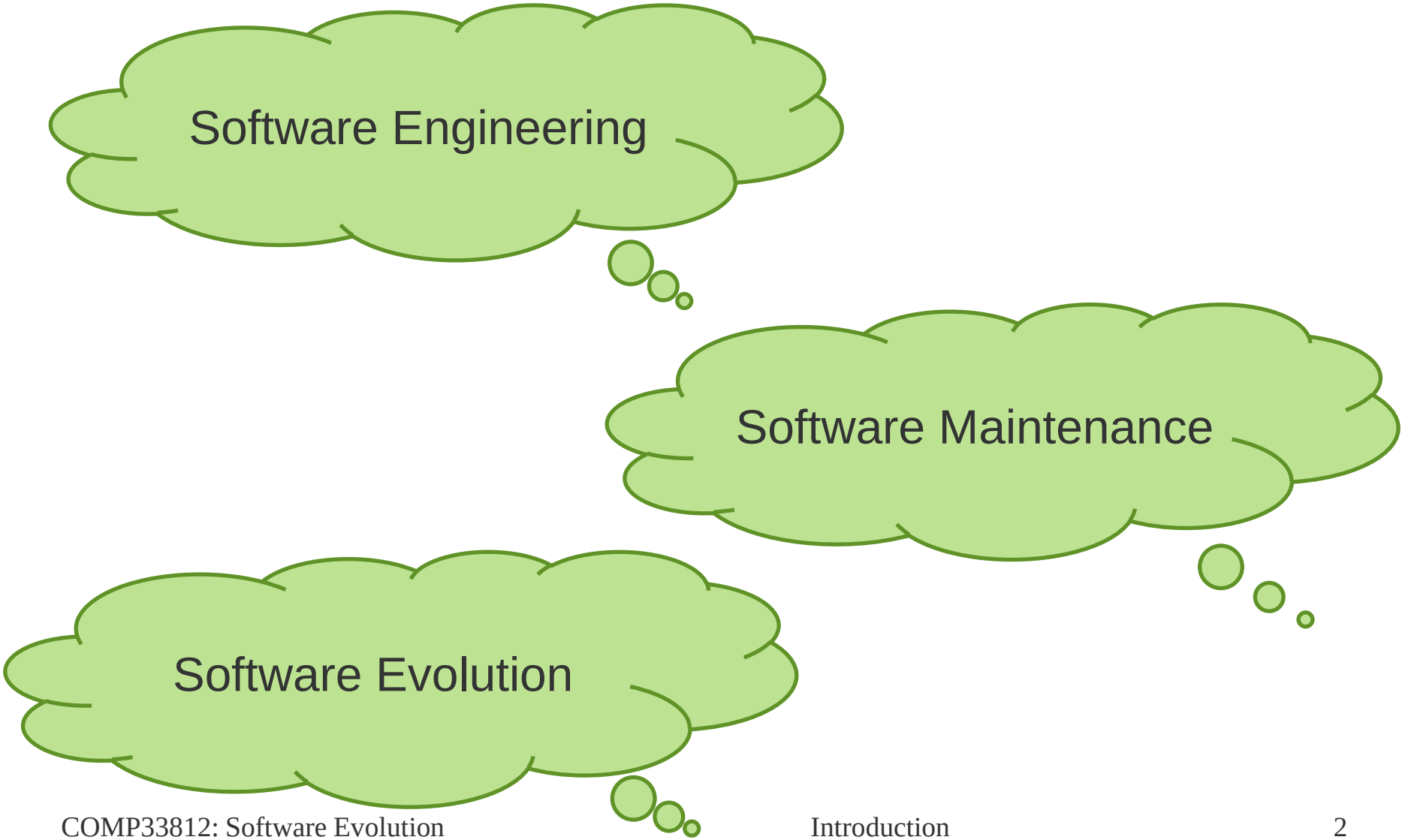
COMP33812: Software Evolution 1

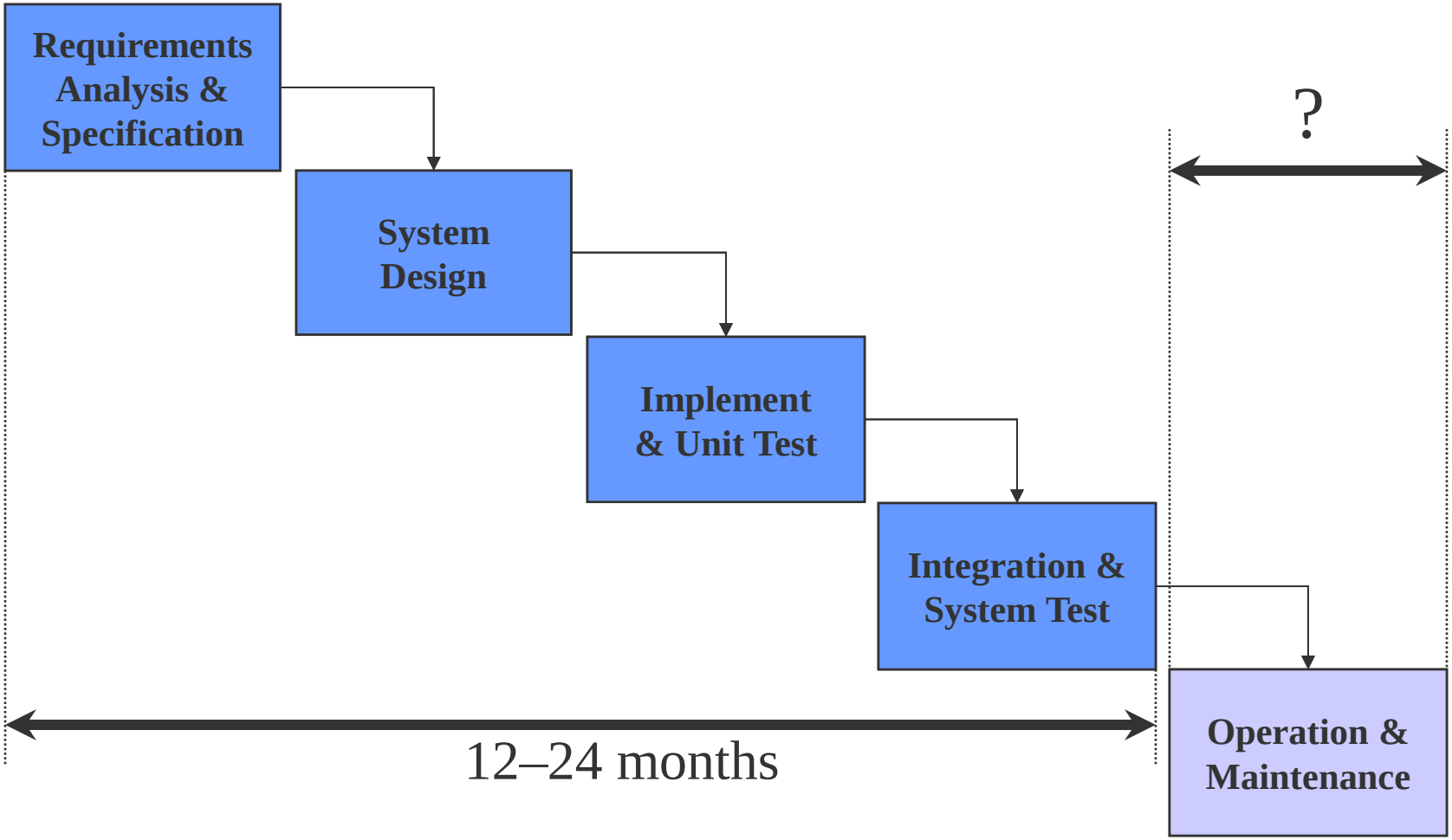
Andy Carpenter
(Andy.Carpenter@manchester.ac.uk)
Sarah Clinch (sarah.clinch@manchester.ac.uk)

Difference?

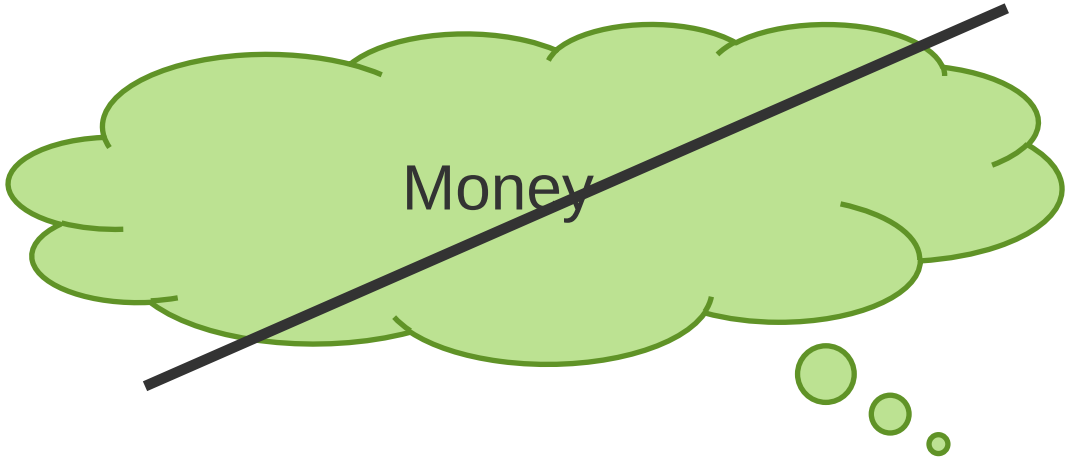

Software Engineering
Software Maintenance
Software Evolution

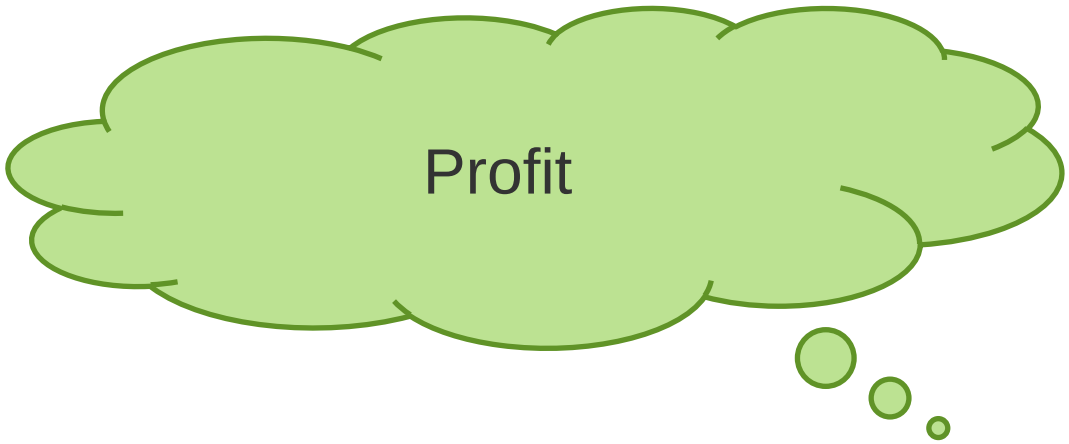
Your Software Evolution Experience?

The Software Lifecycle



Why does maintenance happen?





Why Evolve (Change) Software?

Bug Fix

Corrective change

Enhancement request

Perfective change

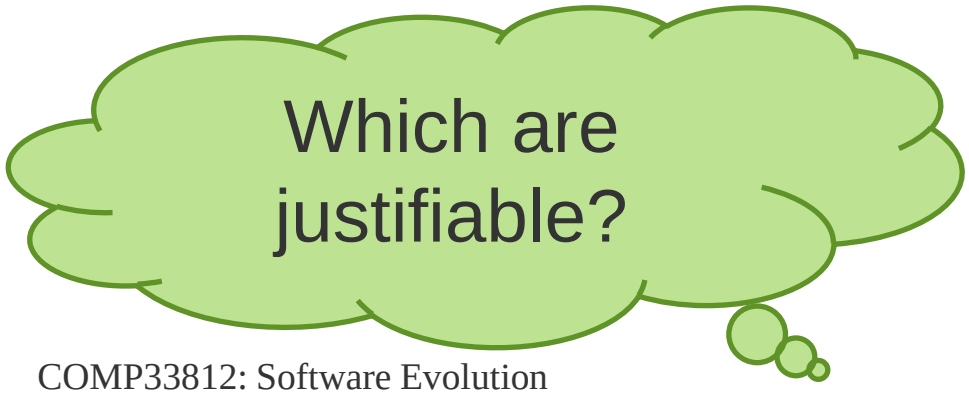
Environmental change

Adaptive change

Refactoring

Preventive change

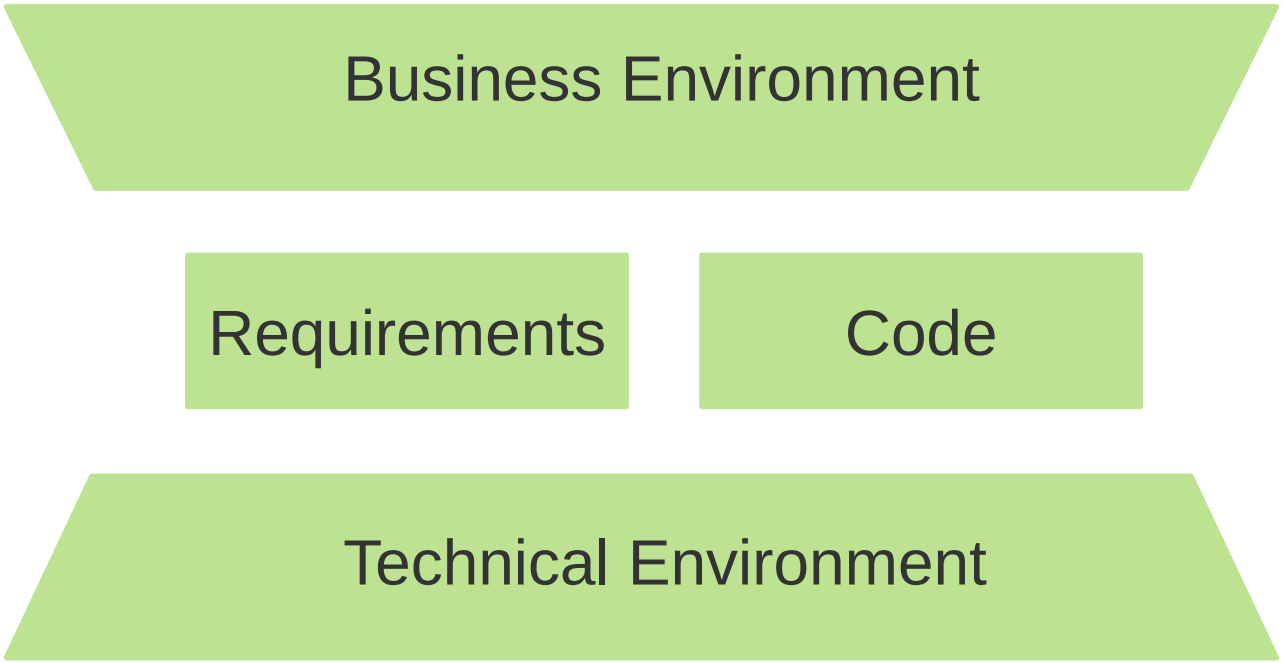
Repay software debt



Types of Maintenance

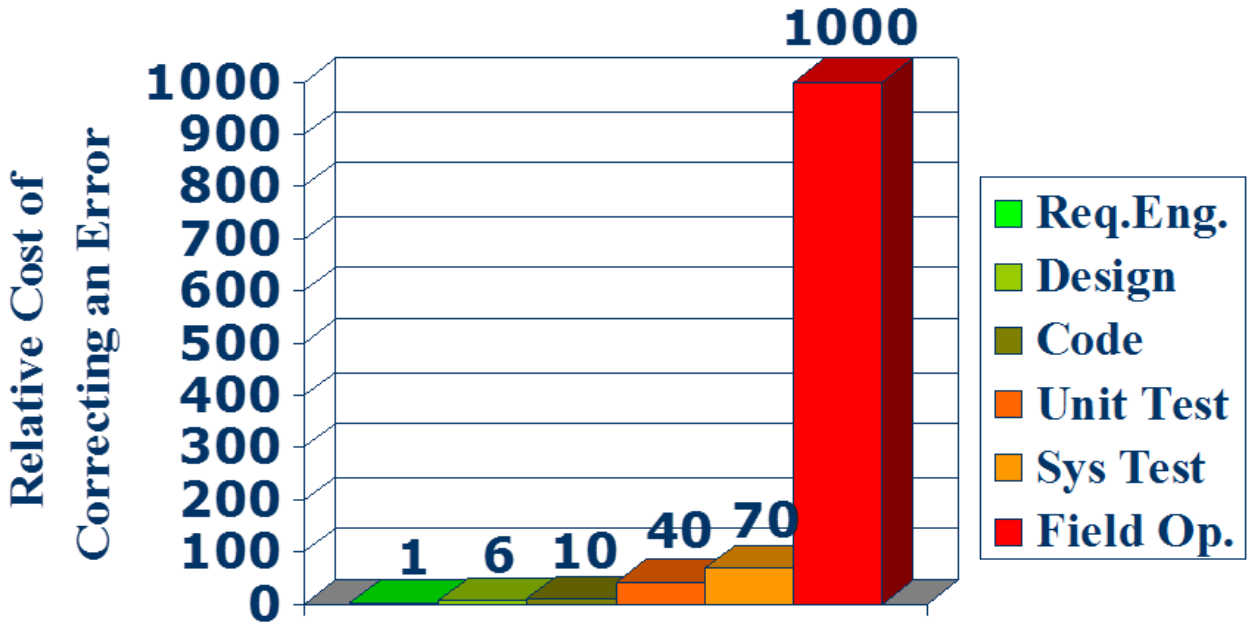
- Corrective change 17%
  - requirements unchanged, code changes
- Adaptive change 18%
  - business or technical environment changes
- Perfective change 65%
  - new stories (enhancement)
- Preventive change 0%
  - refactoring, code cleanup
- Which is the most common?
  - Study by Lientz and Swanson (1980)
- Agile equivalents?

Software and Its Environment



All can change

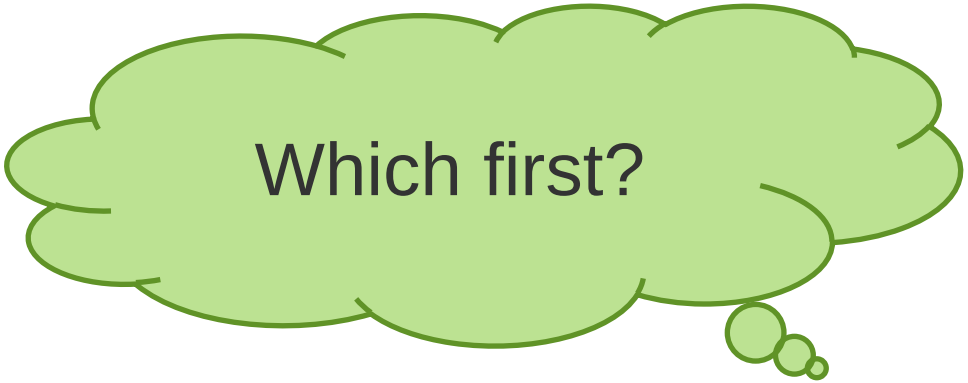
Boehm's Model of Cost of Change



Boehm documented an Air Force project in which the development cost was $30 per line, but the maintenance cost was as high as $4,000 per line [Boehm 75]

Maintain a system – which first?

- Maintaining an e-commerce system
- Perfection requests for:
  - Support promotion of special offers
  - Generate reports for management to monitor business
  - Automatically reorder when stock low



The “Live” System

- While it is operating, the live system is making money for the organisation
- Maintenance activities must be scheduled carefully so that any negative effect is minimised
- E.g. don’t plan to release a new version of the system just before the end of the financial year, when many vital reports are generated
- Plan to make use of quiet periods (e.g. weekends, Xmas vacation)
- Keep mission critical systems functioning!

Software Change is Risky

On Monday, Mr. White's reports all run satisfactorily

But on Tuesday, every *!*?! report exits with an error message!

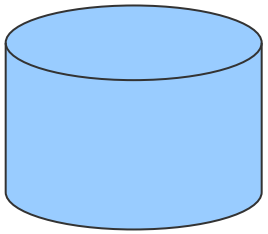
- What happened?

Changes Have Side Effects

- The complexity of real software systems means that changes to one part of the system can have a bad effect on many other parts

CR requests several new reports

Extra data stored in database



Existing reports need to be recompiled/ changed

The Killer Problem is…

- … the hidden (one way) dependency
- Examples of hidden dependencies
  - Variables are dependent on their type
  - Variables are dependent on the expressions they've been involved in previously
  - Files/DBs are dependent on accesses by programs
  - Parameters to procedures are dependent on the caller
  - …

Other examples?

S/W Evolution Affects Profit

- Software Evolution is
  - Difficult
  - Expensive
  - Unavoidable
  - Continuous
- Direct link between the ability of an organisation to change its software systems and its ability to gain and retain its competitive edge.
- Agile techniques/mindset useful for maintenance of non-agile systems, too.

Lehman’s Laws (Background)

- Prof. Meir M. Lehman, who worked at Imperial College London from 1972 to 2002
- Identified a set of behaviours in the evolution of proprietary software
- These behaviours (or observations) are known as Lehman's Laws

Lehman's Laws (First)

"The Law of Continuing Change"

A large program that is used undergoes continuing change or becomes progressively less useful.

The change process continues until it is judged more cost-effective to replace the system with a recreated version."

Lehman's Laws of SE

1. Continuing change
2. Increasing complexity
3. Self regulation
4. Conservation of organisational stability
5. Conservation of familiarity
6. Continuing growth
7. Declining quality
8. Feedback system

What about if Using Agile?

Maintenance is Expensive …

- But is it necessary?
- The solution to the problem of software maintenance is to build systems properly in the first place, right?
- A lesson from 30 years of software engineering:
- Real software systems can never be bug free
- Why?