Software Evolution

Importance of evolution

- Organizations have huge investments in their software systems - they are critical business assets.

- To maintain the value of these assets to the business, they must be changed and updated.

- The majority of the software budget in large companies is devoted to evolving existing software rather than developing new software.

Software Change

- Software change is inevitable
  - New requirements emerge when the software is used
  - The business environment changes
  - Errors must be repaired
  - New computers and equipment is added to the system
  - The performance or reliability of the system may have to be improved.

- A key problem for organizations is implementing and managing change to their existing software systems.

Software Maintenance

> "Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment."
>
> ANSI/IEEE Standard 729-1983

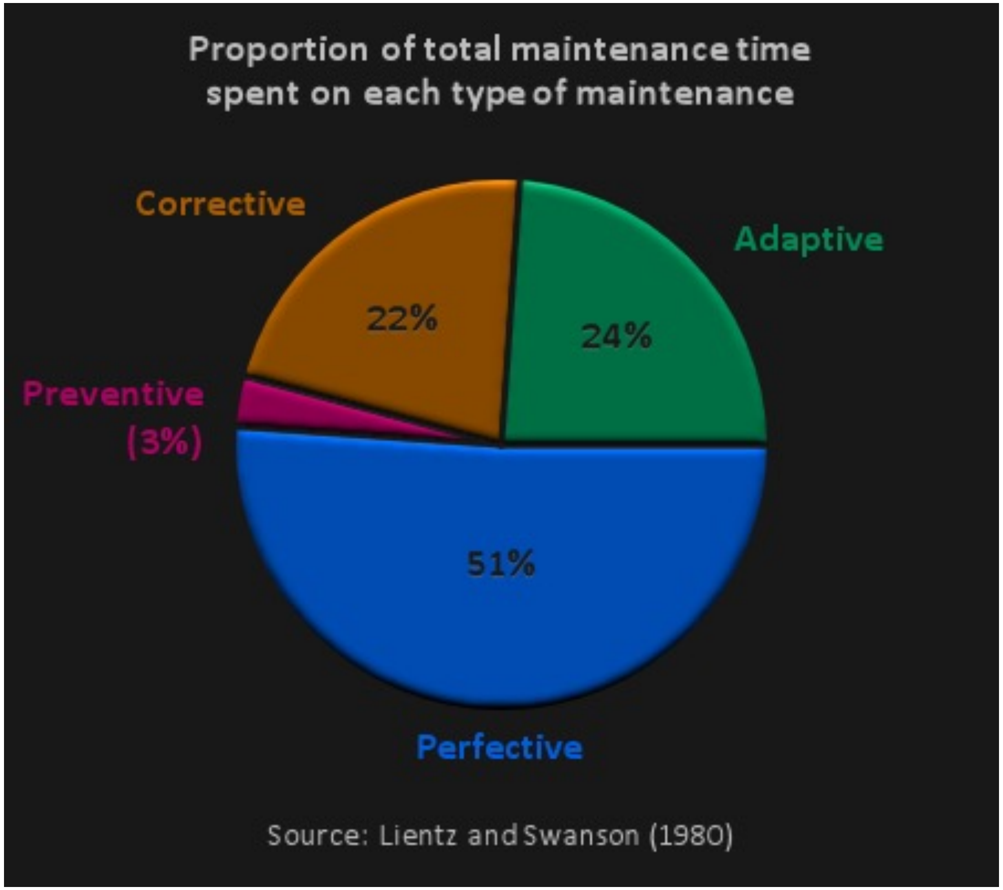
Software Maintenance

- Modifying a program after it has been put into use.
- Maintenance does not normally involve major changes to the system’s architecture.
- Changes are implemented by modifying existing components and adding new components to the system.

Software Maintenance (ISO/IEC standard)

- Perfective Maintenance: Any modification of a software product after delivery to improve performance or maintainability
- Corrective Maintenance: Reactive modification of a software product performed after delivery to correct discovered faults
- Adaptive Maintenance: Modification of a software product performed after delivery to keep a computer program usable in a changed or changing environment
- Preventive Maintenance: Software modifications performed for the purpose of preventing problems before they occur

Software Maintenance time

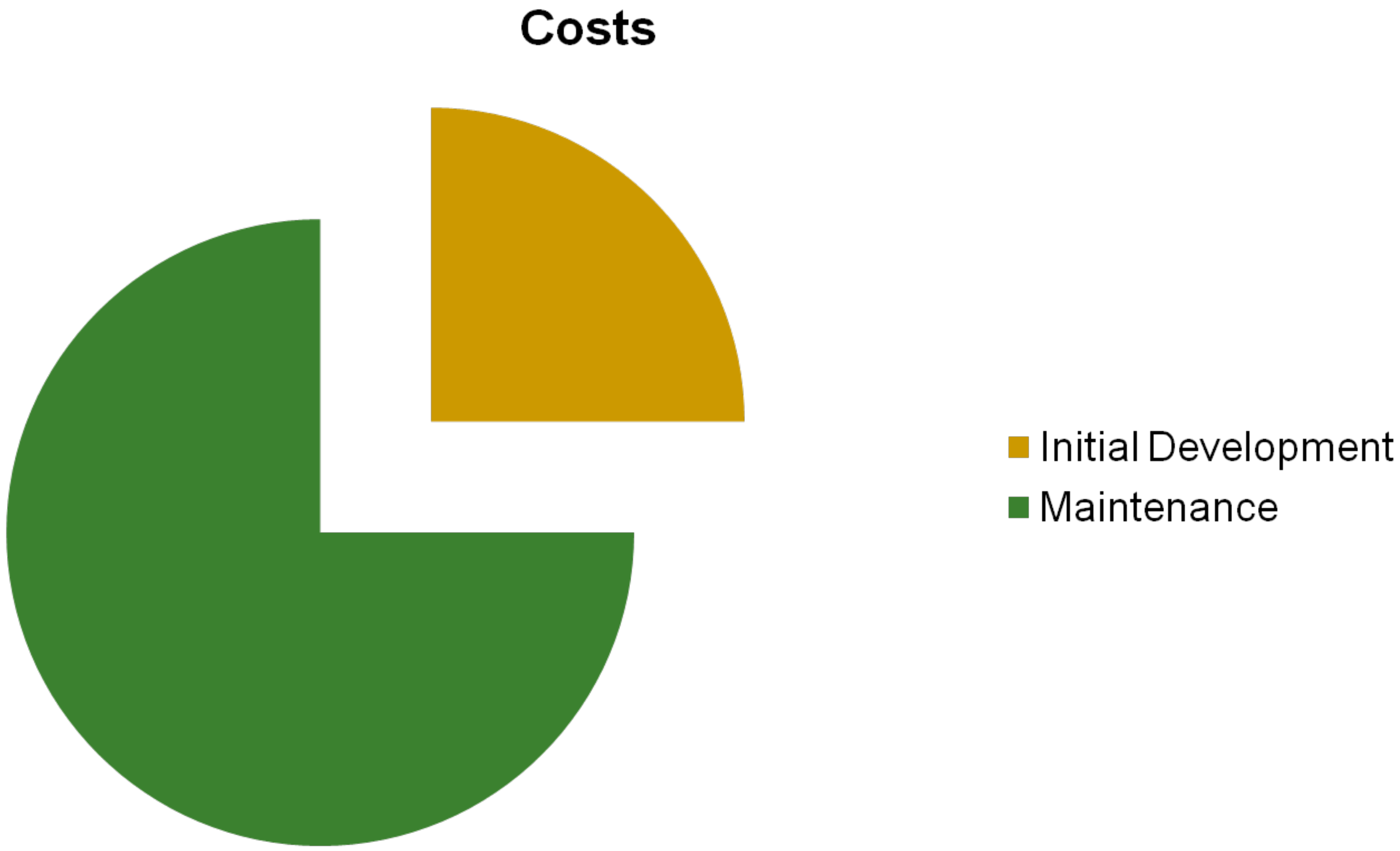


Source: Lientz and Swanson (1980)

Maintenance costs

- Usually greater than development costs (2* to 100* depending on the application).
- Affected by both technical and non-technical factors.
- Increases as software is maintained. Maintenance corrupts the software structure so makes further maintenance more difficult.
- Ageing software can have high support costs (e.g. old languages, compilers etc.).

Development/maintenance costs



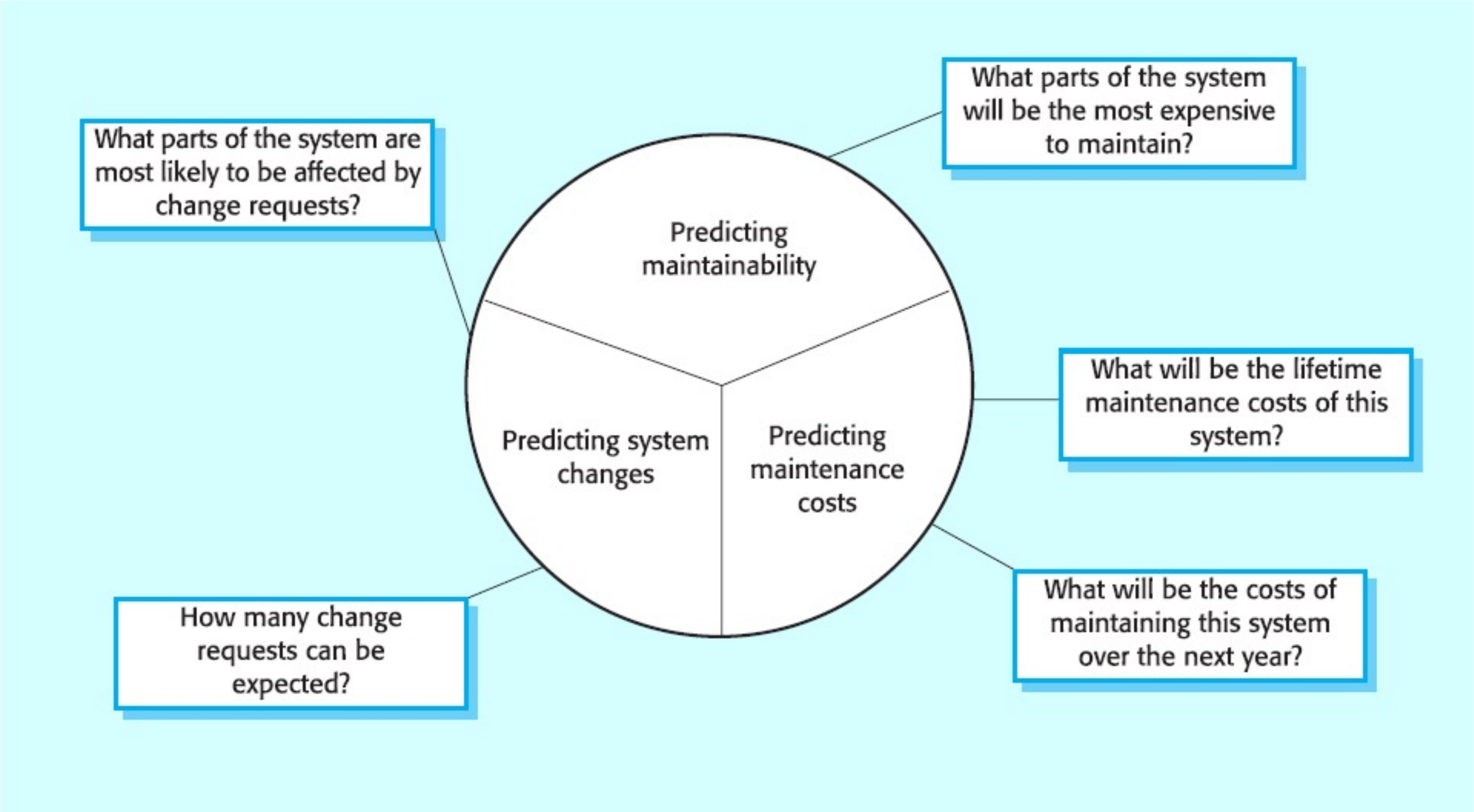
Maintenance cost factors

- Team stability
  - Maintenance costs are reduced if the same staff are involved with them for some time.
- Contractual responsibility
  - The developers of a system may have no contractual responsibility for maintenance so there is no incentive to design for future change.
- Staff skills
  - Maintenance staff are often inexperienced and have limited domain knowledge.
- Program age and structure
  - As programs age, their structure is degraded and they become harder to understand and change.

Maintenance prediction

- Maintenance prediction is concerned with assessing which parts of the system may cause problems and have high maintenance costs
  - Change acceptance depends on the maintainability of the components affected by the change;
  - Maintenance costs depend on the number of changes and costs of change depend on maintainability.

Maintenance prediction


What parts of the system are most likely to be affected by change requests?
What parts of the system will be the most expensive to maintain?
Predicting maintainability
What will be the lifetime maintenance costs of this system?
Predicting system changes
Predicting maintenance costs
How many change requests can be expected?
What will be the costs of maintaining this system over the next year?

Change Prediction

- Predicting the number of changes requires and understanding of the relationships between a system and its environment.
- Tightly coupled systems require changes whenever the environment is changed.
- Factors influencing this relationship are
  - Number and complexity of system interfaces
  - Number of inherently volatile system requirements
  - The business processes where the system is used

Complexity metrics

- Predictions of maintainability can be made by assessing the complexity of system components.
- Studies have shown that most maintenance effort is spent on a relatively small number of system components.
- Complexity depends on
  - Complexity of control structures
  - Complexity of data structures
  - Object, method (procedure) and module size

Process metrics

- Process measurements may be used to assess maintainability
  - Number of requests for corrective maintenance
  - Average time required for impact analysis
  - Average time taken to implement a change request
  - Number of outstanding change requests

Let’s digress at bit and talk about how changing software works in industry

- Adding a feature
- Fixing a bug
- Improving the design
- Optimizing resource usage

Adding Feature and Fixing Bugs

Changing Logo on a web site from left to right

- Is it adding a feature or fixing a bug?

```
public class CDPlayer
{
   public void addTrackListing(Track track)
   {
      …
    }
}
```

```
public class CDPlayer
{
   public void addTrackListing(Track track)
   {
      …
    }
  public void replaceTrackListing(String name, Track track)
   {
      …
    }

}
```

Changing Software

	Adding a Feature	Fixing a Bug	Refactoring	Optimizing
Structure	Changes	Changes	Changes	----
Functionality	Changes	Changes	----	----
Resource Usage	----	----	----	Changes

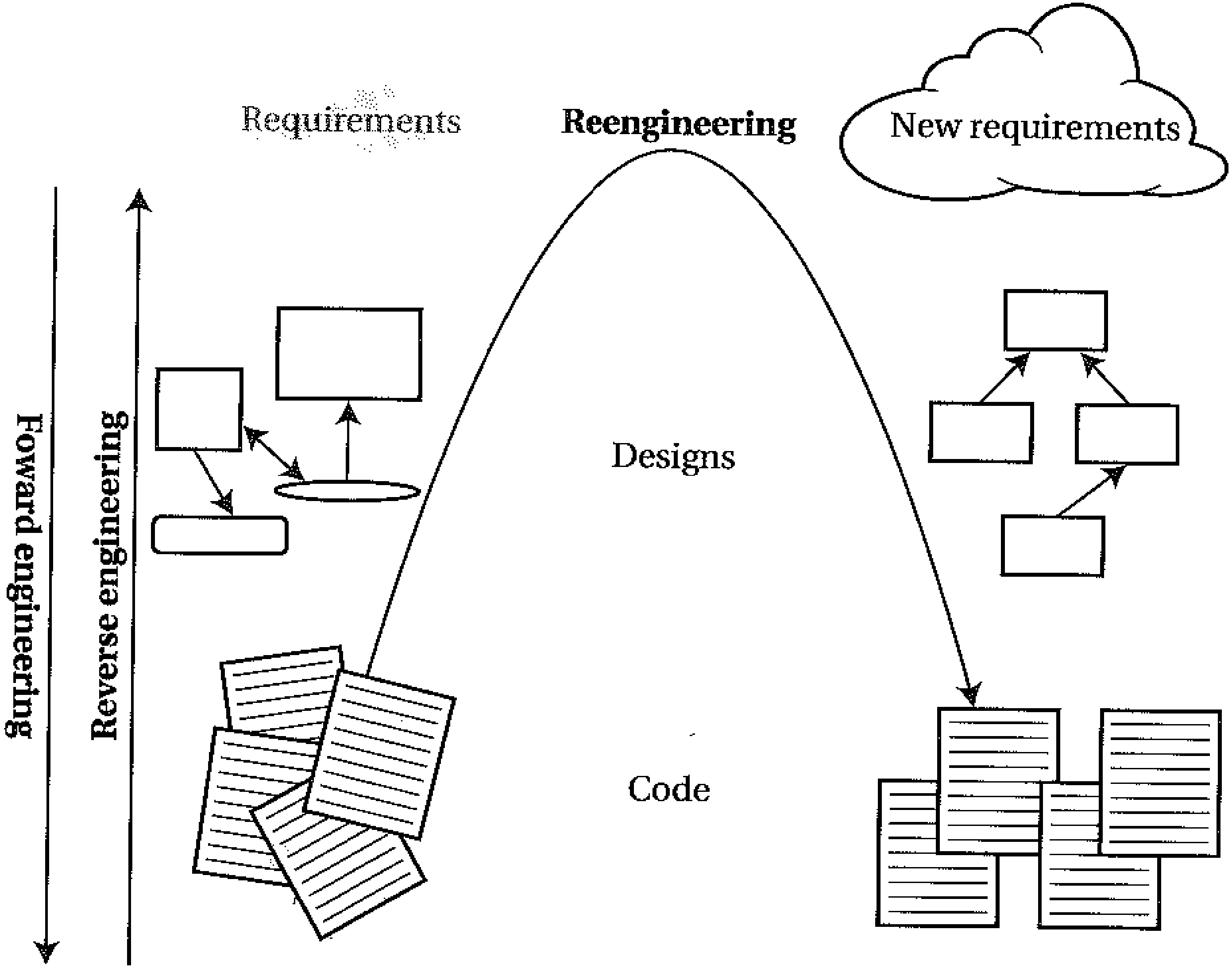
Changes in Systems

- Changes in a system are made in two primary ways
  - Edit and Pray
  - Cover and Modify

Example change process

- Identify change points
- Find test points
- Break dependencies
- Write tests
- Make changes and refactor

Forward, Reverse, Re-engineering



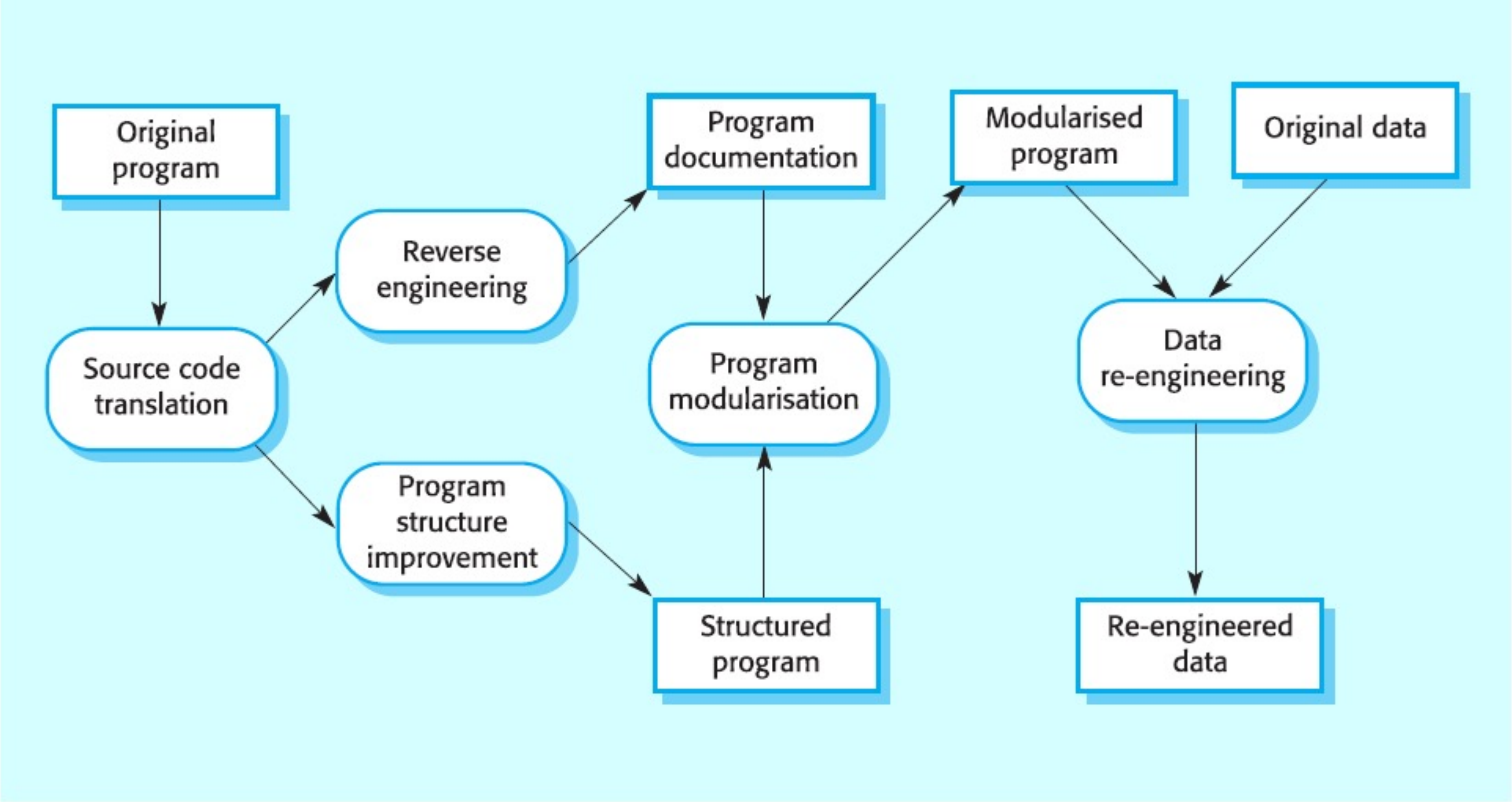
Why do we Reengineer?

- What is Legacy Software?
  - It may not be that old
- Goal of Reengineering is to reduce the complexity of legacy system sufficiently that it can be continue to be used and adapted at an acceptable cost
- Why Reengineer?
  - Unbundle a monolithic system
  - Might need an improvement in performance
  - Port the system to a new platform/technology
  - Extract the design – enables a new implementation
  - Reduce human dependencies

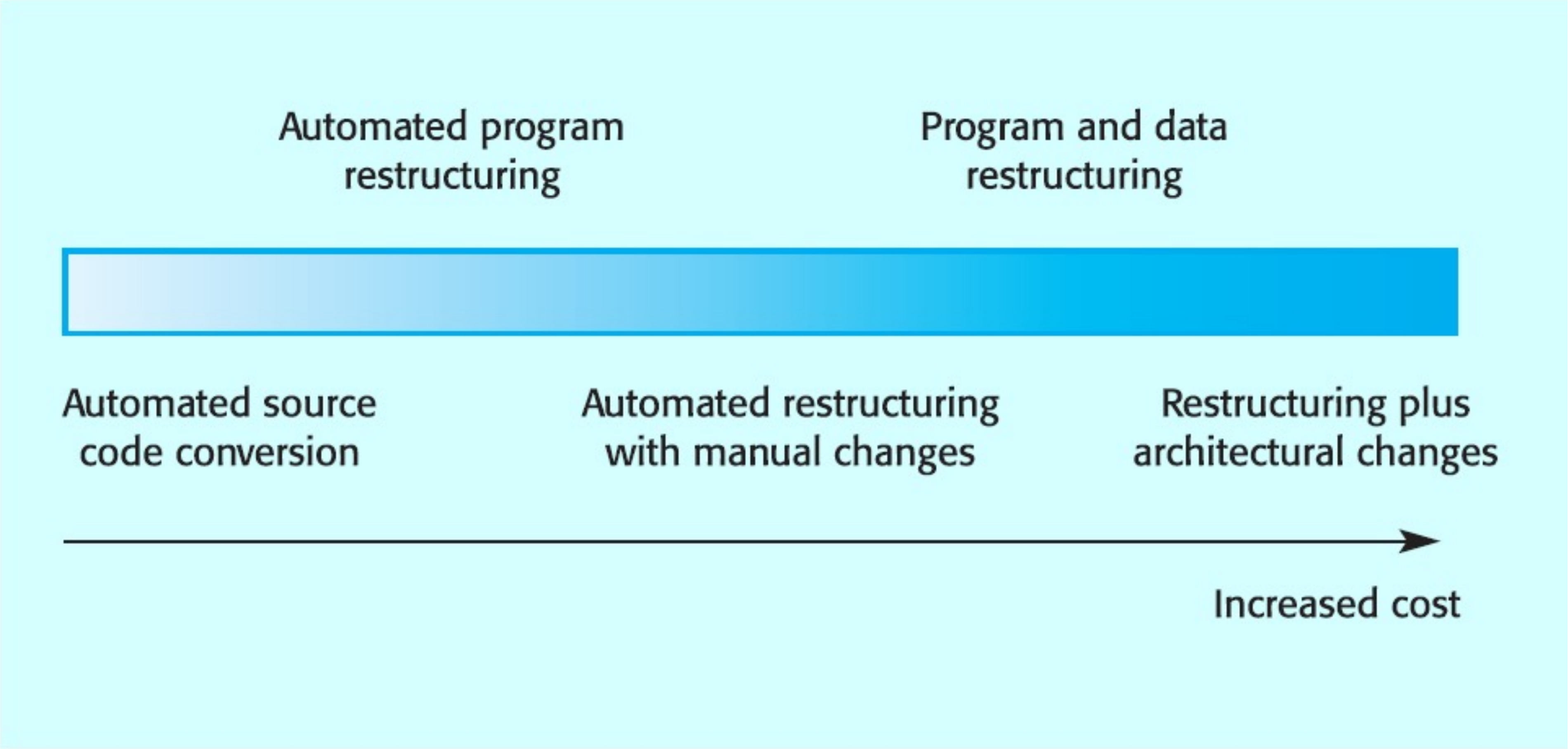
When do we need to Reengineer?

- Obsolete or no documentation
- Missing tests
- Departure of original developers or users
- Disappearance of inside knowledge about the system
- Limited understanding of the entire system
- Too long to turn things over to production
- Too much time to make simple changes
- Need for constant bug fixes
- Big build times
- Difficulties separating products
- Duplicated code
- Code Smells

The re-engineering process



Re-engineering approaches


Automated program restructuring
Program and data restructuring
Automated source code conversion
Automated restructuring with manual changes
Restructuring plus architectural changes
Increased cost

Re-engineering cost factors

- The quality of the software to be reengineered.
- The tool support available for reengineering.
- The extent of the data conversion which is required.
- The availability of expert staff for reengineering.
  - This can be a problem with old systems based on technology that is no longer widely used.