Software Engineering

Lesson #08 - Lecture

Lesson #08 - Lecture

Your KBTU 202309 Software Engineering class information is updating …

Lesson #08 update is in progress

This will take around 2 hours to complete

Please, don’t turn off your head

Software Evaluation

Software Evolution

Software Evaluation

```
#include<iostream>
Using namespace std;

int main()
{
      cout << “Software Evolution” << endl;

      return 0;
}
```

Agenda: Lesson #08 - Software Engineering - Lecture

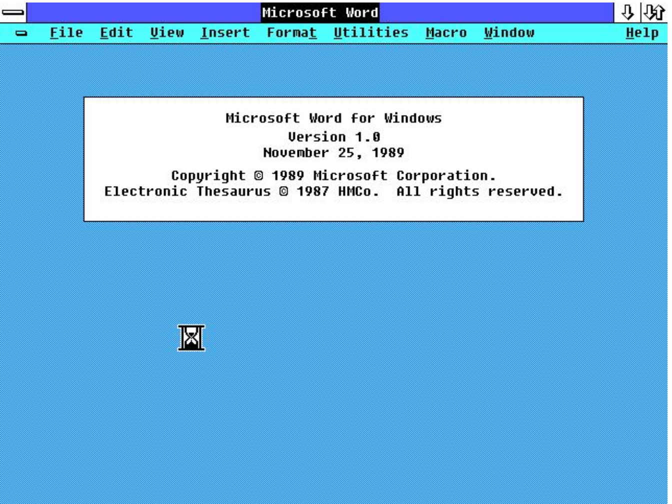
Evolution processes

Large software systems usually have a long lifetime

For example, military or infrastructure systems, such as air traffic control systems, may have a lifetime of 30 years or more. Business systems are often more than 10 years old

Evolution processes

The first version of Microsoft Word was introduced in 1983, so it has been around for more than 40 years



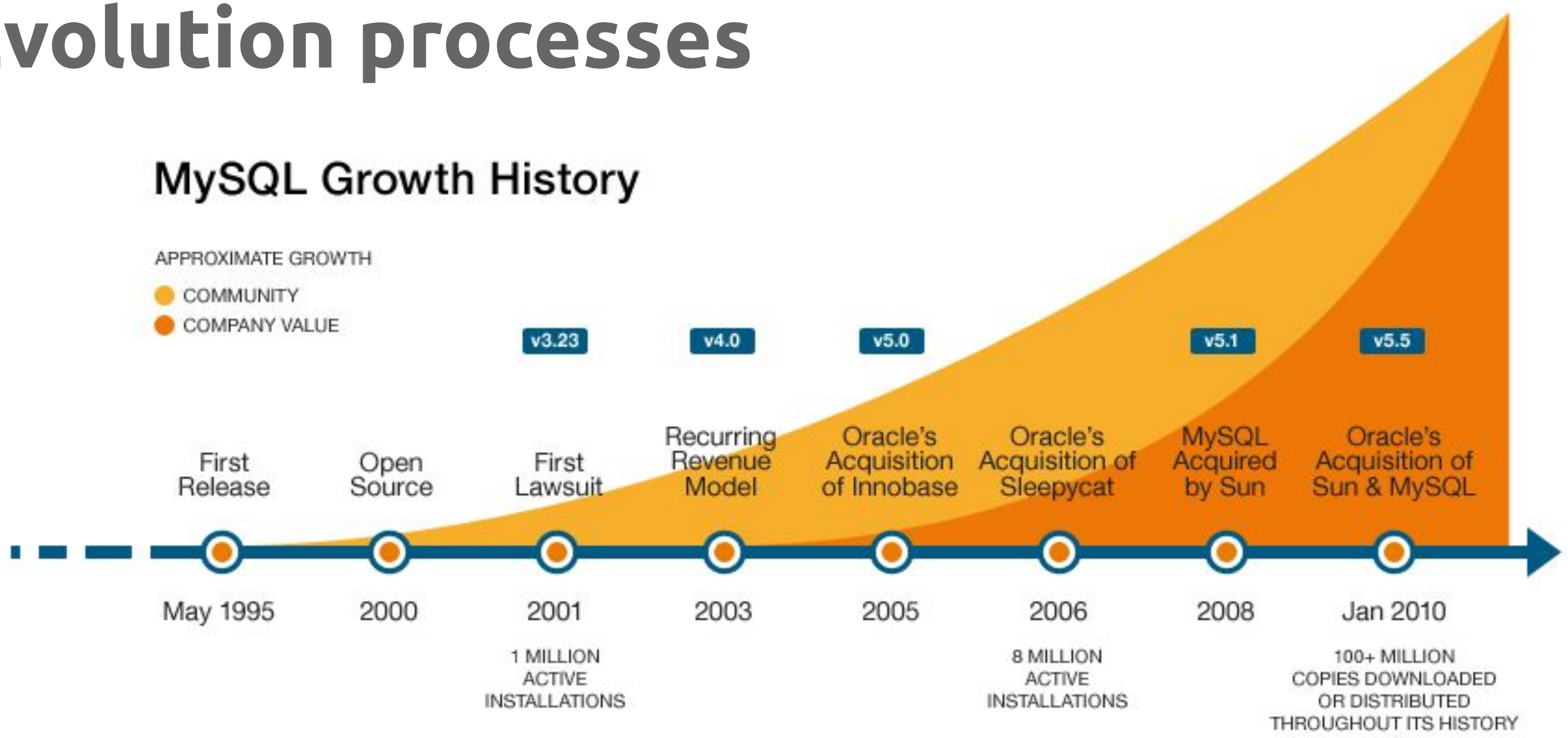
Microsoft Word for Windows
Version 1.0
November 25, 1989

Copyright © 1989 Microsoft Corporation.
Electronic Thesaurus © 1987 HMCo. All rights reserved.

More than 40 years

Evolution processes

Evolution processes



History of Oracle Database Versions

	acle 19c	Automated testing of query plans, Automatic indexing
	Oracle 18c	Polymorphic Table Functions, Active Directory Integration
2013	Oracle 12c	Cloud plugging
2007	Oracle 11g	Manageability, diagnosability and availability
2004	Oracle 10g	Grid computing
2001	Oracle 9i	Oracle RAC
1999	Oracle 8i	Internet computing
1997	Oracle 8	Objects and partitioning
1992	Oracle 7	PL/SQL stored program
1988	Oracle 6	disk I/O, row locking, scalability and backup and
1985	Oracle 5	client/server computing and distributed da
1984	Oracle 4	multiversion read consistency
1983	Oracle 3	first relational database to ru
1979	Oracle 2	SQL based RDBMS
1978	Oracle 1	never officially re

ORACLE EVOLUTION

2016

Oracle v1: on-chain request, off-chain provider; example Oraclize

2019

Oracle v2: on-chain provider; example Chainlink

2022

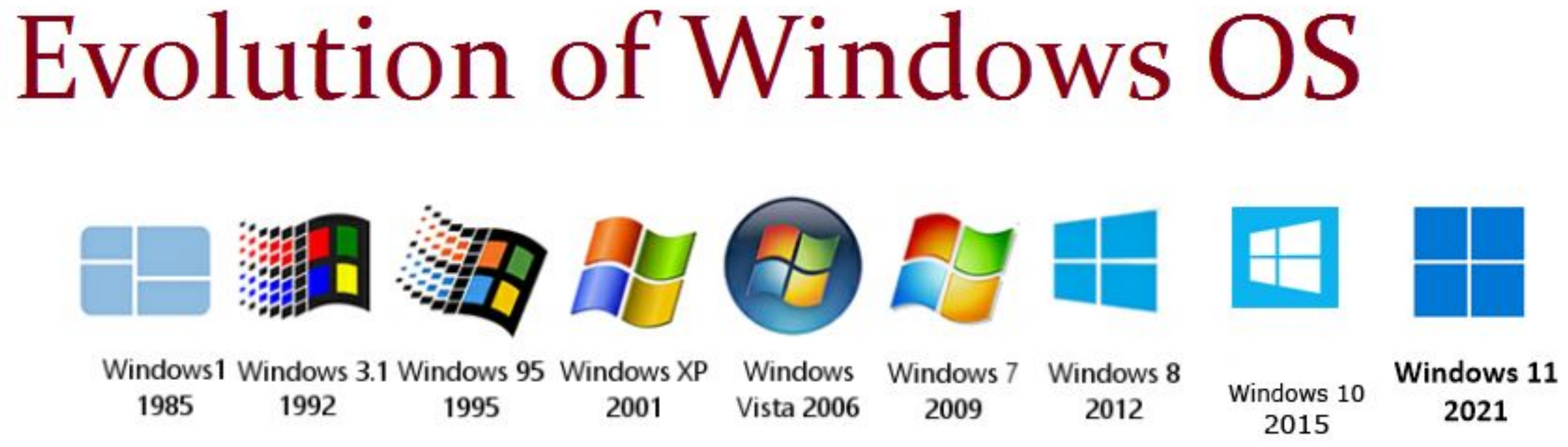
Oracle v3: off-chain data, on-chain verifier, example Chainlink (in alpha)

2023 - 2025

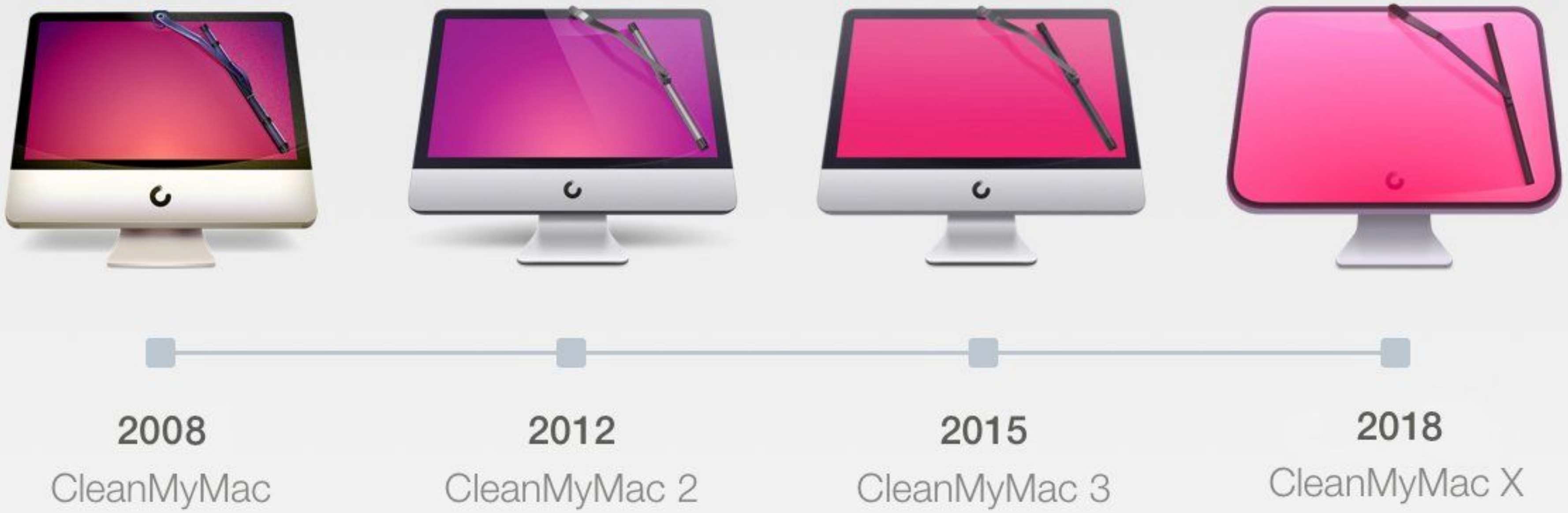
Oracle v4: zero knowledge provable data

2025+

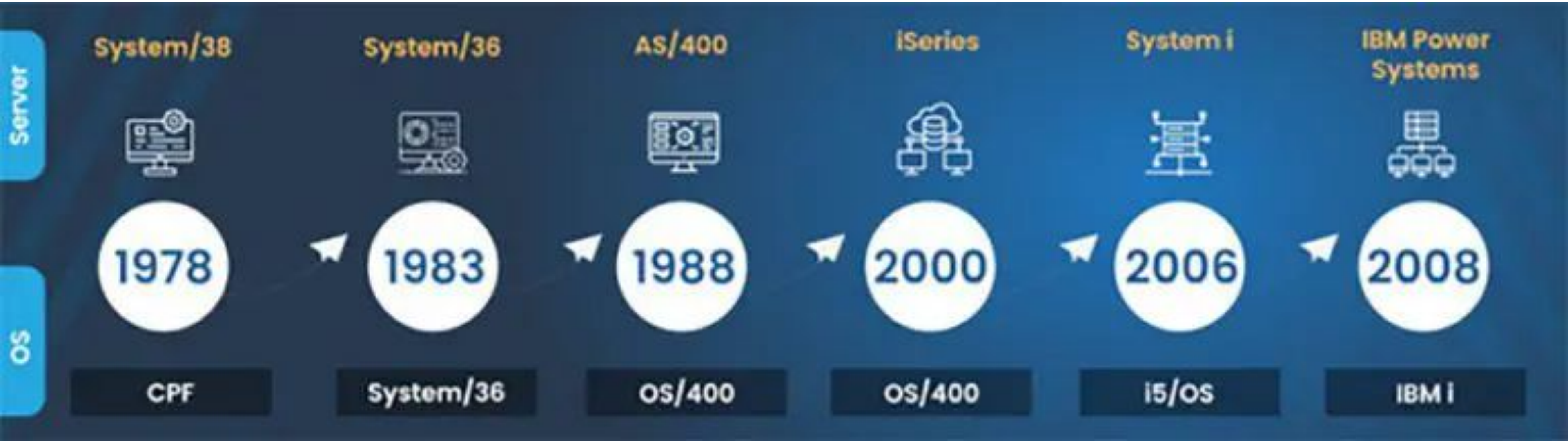
Evolution processes



Evolution processes



Evolution processes



Server	System/38	System/36	AS/400	iSeries	System i	IBM Power Systems
	1978	1983	1988	2000	2006	2008
OS	CPF	System/36	OS/400	OS/400	i5/OS	IBM i

Evolution processes

Businesses have to change their software to ensure that they continue to get value from it

Their systems are critical business assets, and they have to invest in change to maintain the value of these assets

Evolution processes

The requirements of installed software systems change as the business and its environment change, so new releases of the systems that incorporate changes and updates are usually created at regular intervals

Software engineering is therefore a spiral process with requirements, design, implementation, and testing going on throughout the lifetime of the system

Software change

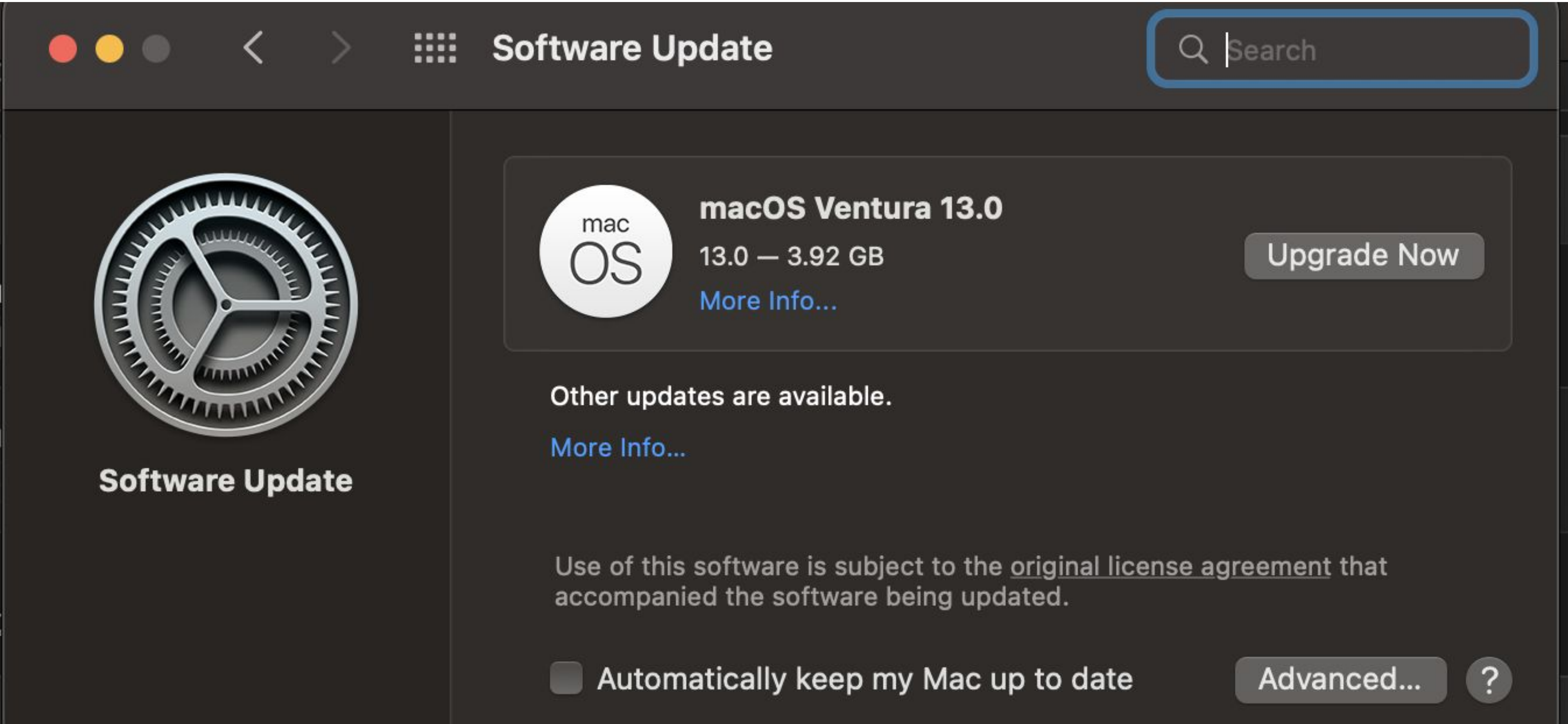
Software change is inevitable

- 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

Software change

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

Evolution processes



macOS Ventura 13.0

13.0 — 3.92 GB

More Info...

Upgrade Now

Other updates are available.

More Info...

Use of this software is subject to the original license agreement that accompanied the software being updated.

Automatically keep my Mac up to date

Advanced...

?

Evolution processes

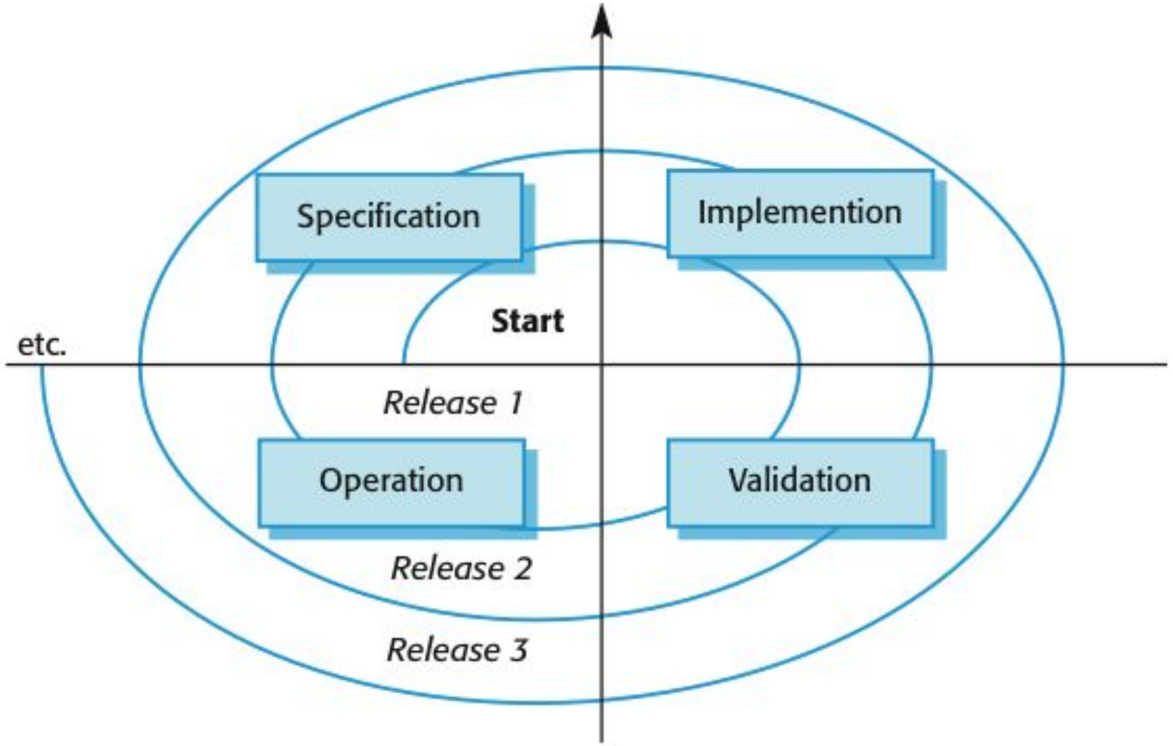
Importance of evolution

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

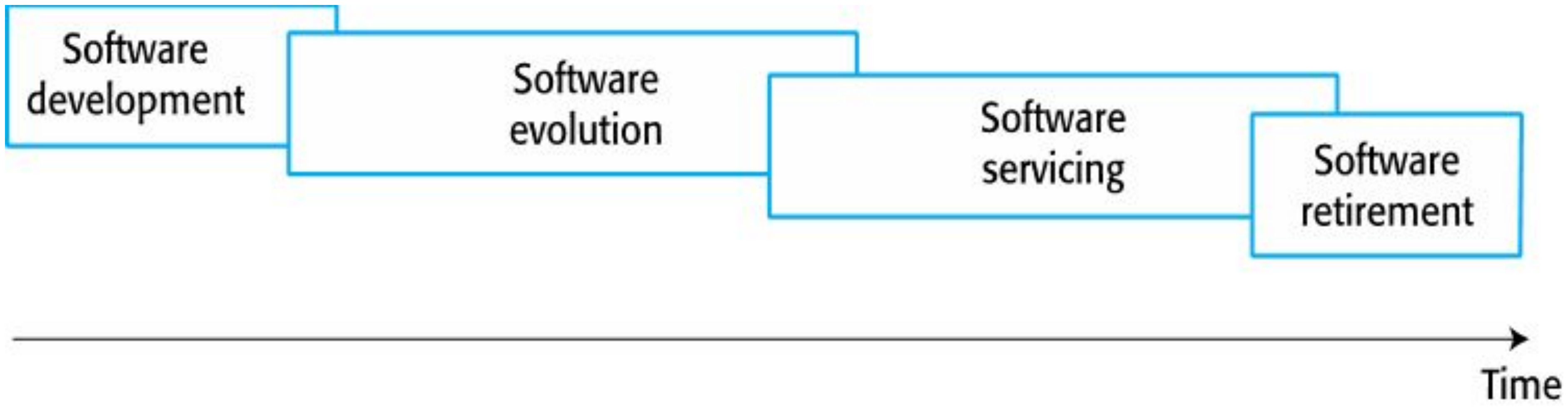
To maintain the value of these assets to the business, they must be changed and updated

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

A spiral model of development and evolution

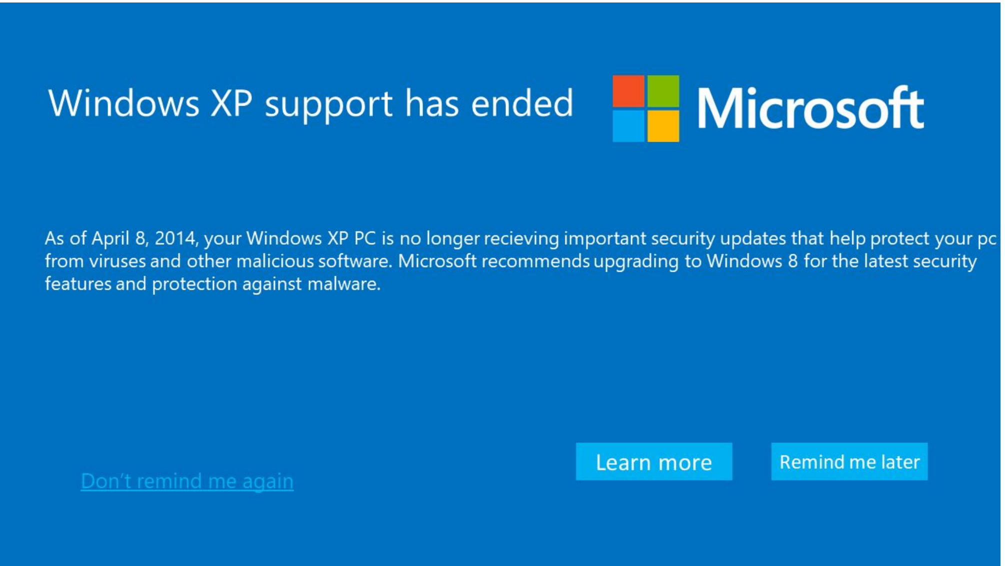


Evolution and servicing



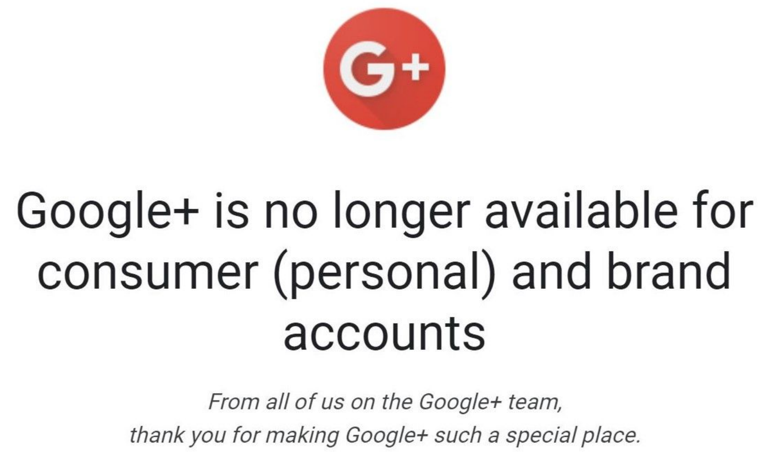
Evolution processes

Evolution and servicing


Windows XP support has ended

Microsoft

As of April 8, 2014, your Windows XP PC is no longer recieving important security updates that help protect your pc from viruses and other malicious software. Microsoft recommends upgrading to Windows 8 for the latest security features and protection against malware.


Google+ is no longer available for consumer (personal) and brand accounts

From all of us on the Google+ team,
thank you for making Google+ such a special place.

Evolution and servicing

Evolution

- The stage in a software system's life cycle where it is in operational use and is evolving as new requirements are proposed and implemented in the system

Evolution and servicing

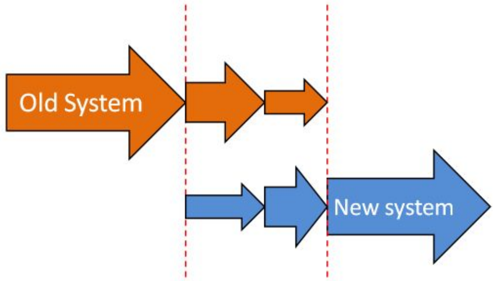
Servicing

- At this stage, the software remains useful but the only changes made are those required to keep it operational i.e. bug fixes and changes to reflect changes in the software’s environment. No new functionality is added.

Evolution and servicing

Phase-out

- The software may still be used but no further changes are made to it

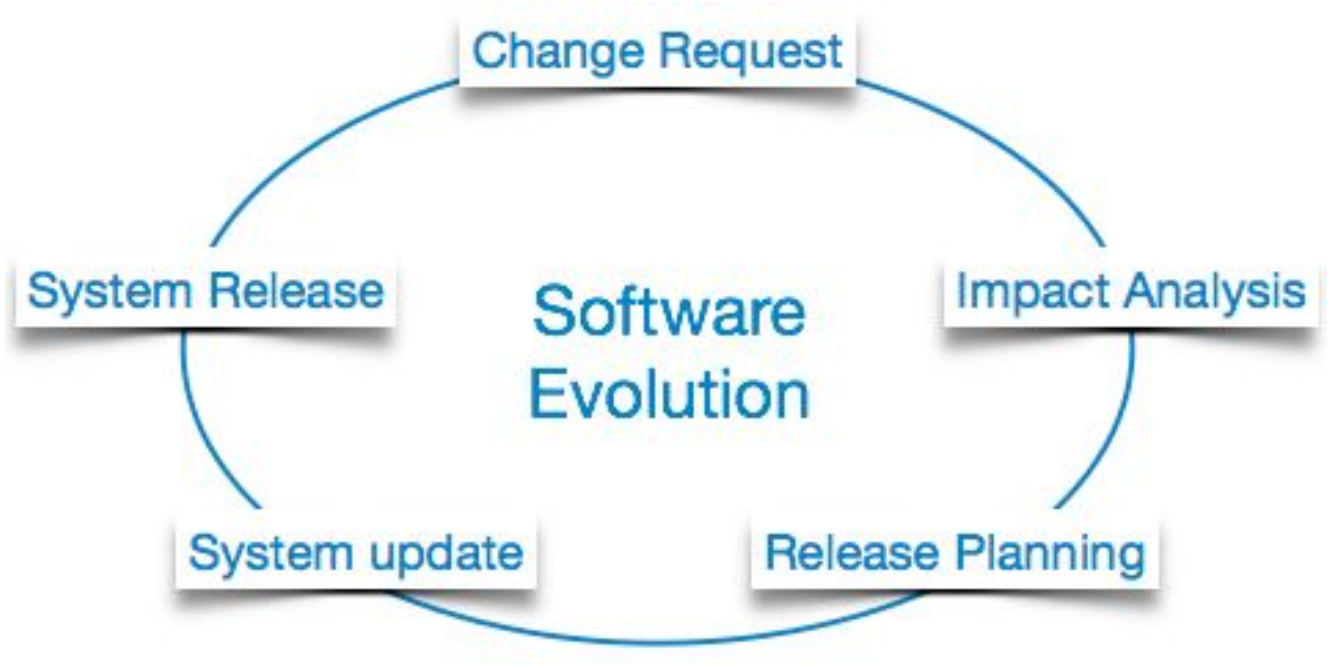


Evolution processes

As with all software processes, there is no such thing as a standard software change or evolution process

The most appropriate evolution process for a software system depends on the type of software being maintained, the software development processes used in an organization, and the skills of the people involved

Evolution processes



Agenda: Lesson #08 - Software Engineering - Lecture

1	**Evolution processes**
2	Legacy systems
3	Software maintenance
4	Q & A

Evolution processes

Software evolution processes depend on

- The type of software being maintained;
- The development processes used;
- The skills and experience of the people involved

Evolution processes

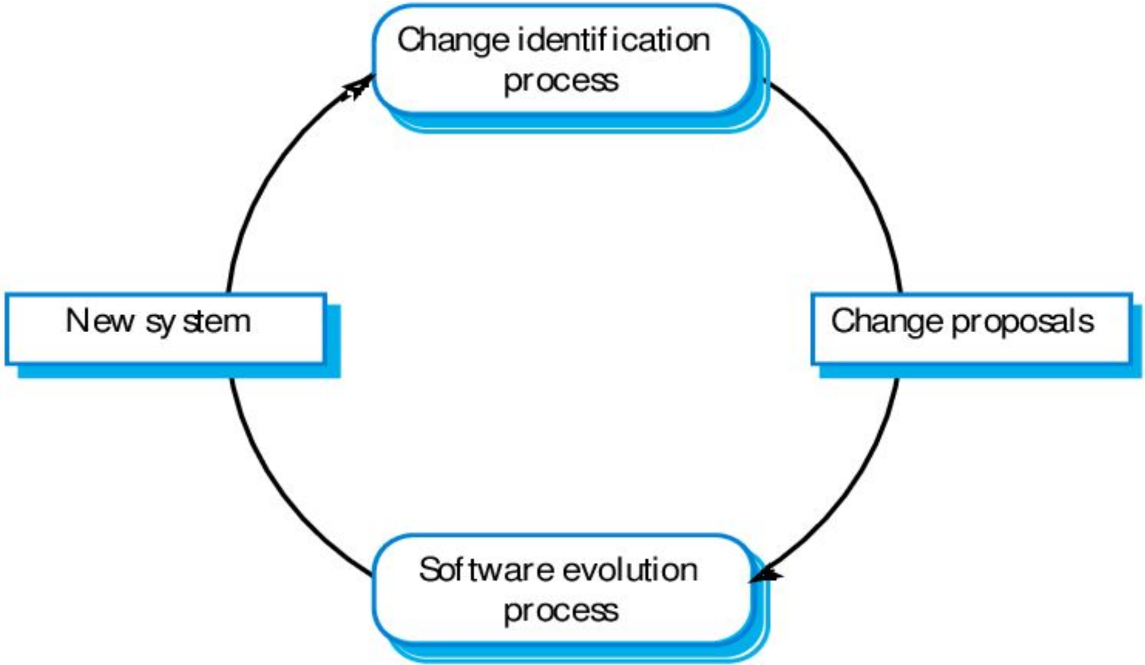
Proposals for change are the driver for system evolution

- Should be linked with components that are affected by the change, thus allowing the cost and impact of the change to be estimated

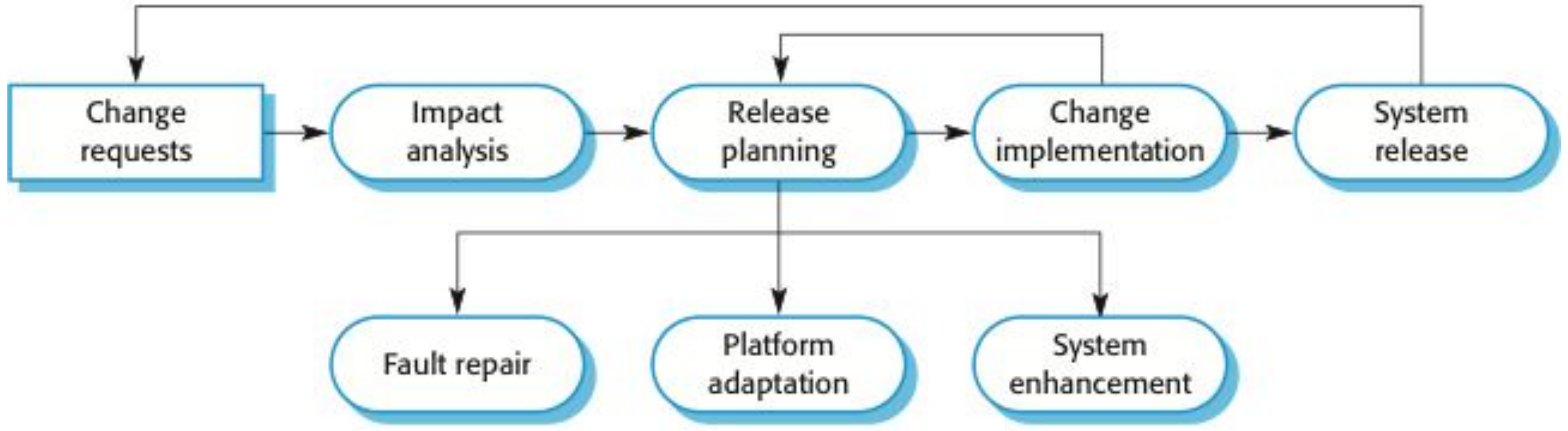
Change identification and evolution continues throughout the system lifetime

Evolution processes

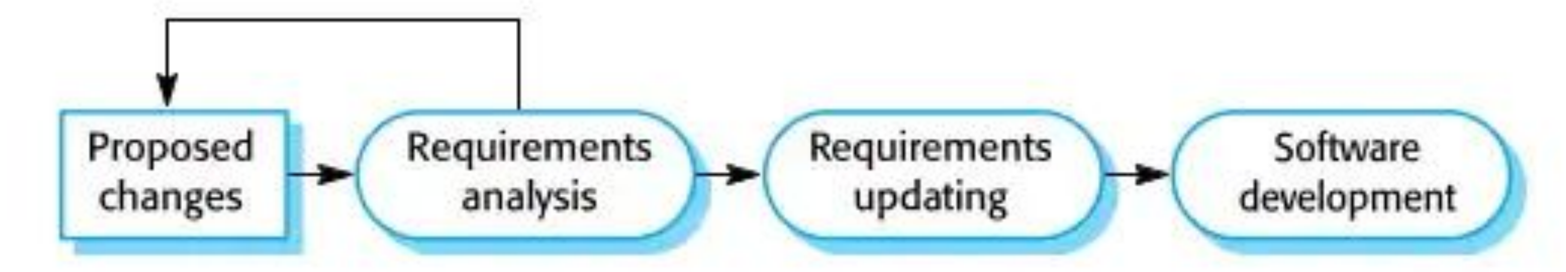
The processes of change identification and system evolution are cyclical and continue throughout the lifetime of a system



The software evolution process



Change implementation



Change implementation

Iteration of the development process where the revisions to the system are designed, implemented and tested

A critical difference is that the first stage of change implementation may involve program understanding, especially if the original system developers are not responsible for the change implementation

Change implementation

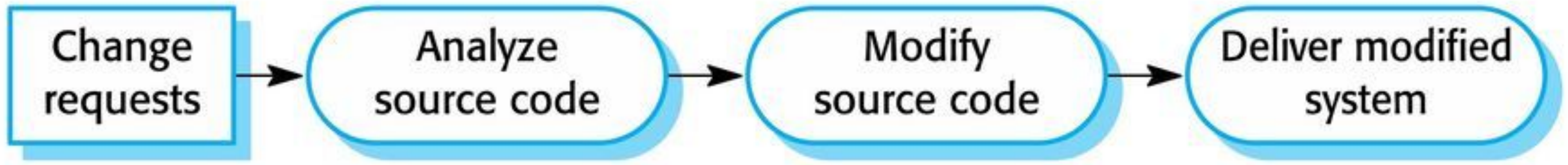
During the program understanding phase, you have to understand how the program is structured, how it delivers functionality and how the proposed change might affect the program

Urgent change requests

Urgent changes may have to be implemented without going through all stages of the software engineering process

- If a serious system fault has to be repaired to allow normal operation to continue;
- If changes to the system’s environment (e.g. an OS upgrade) have unexpected effects;
- If there are business changes that require a very rapid response (e.g. the release of a competing product)

The emergency repair process



Agile methods and evolution

Agile methods are based on incremental development so the transition from development to evolution is a seamless one

- Evolution is simply a continuation of the development process based on frequent system releases

Automated regression testing is particularly valuable when changes are made to a system
Changes may be expressed as additional user stories

Handover problems

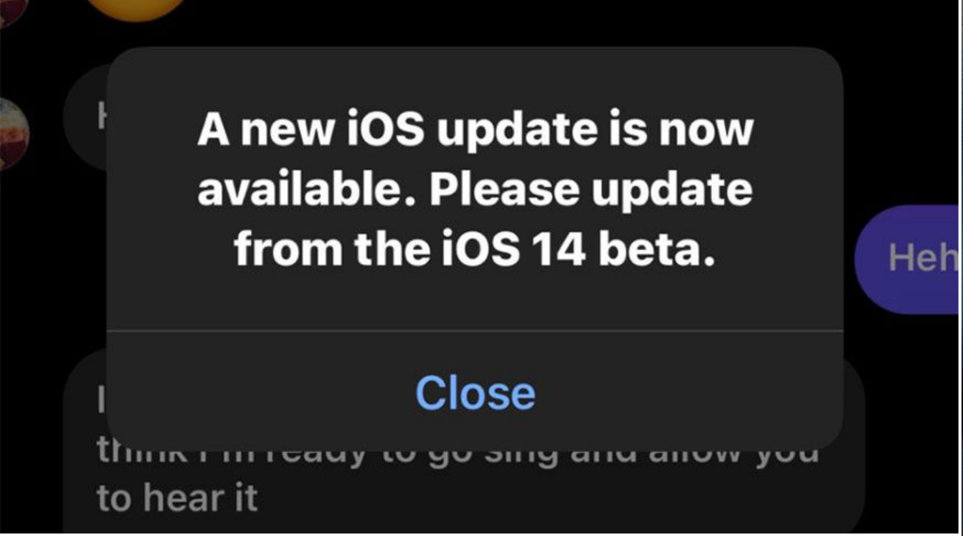
Where the development team have used an agile approach but the evolution team is unfamiliar with agile methods and prefer a plan-based approach

- The evolution team may expect detailed documentation to support evolution and this is not produced in agile processes

Handover problems

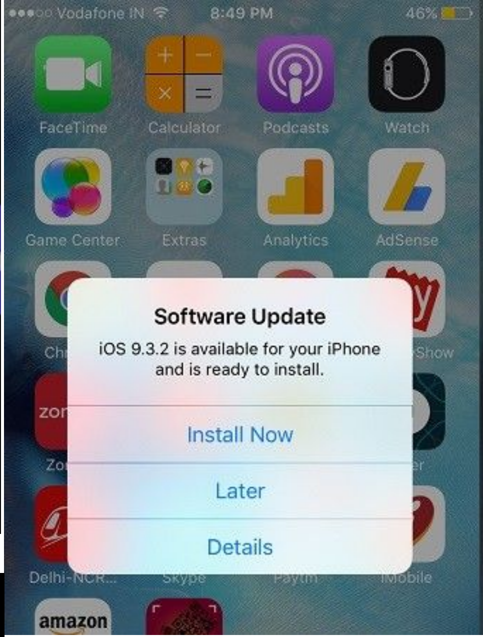
Where a plan-based approach has been used for development but the evolution team prefer to use agile methods

- The evolution team may have to start from scratch developing automated tests and the code in the system may not have been refactored and simplified as is expected in agile development


A new iOS update is now available. Please update from the iOS 14 beta.

Close

think I'm ready to go sing and allow you to hear it

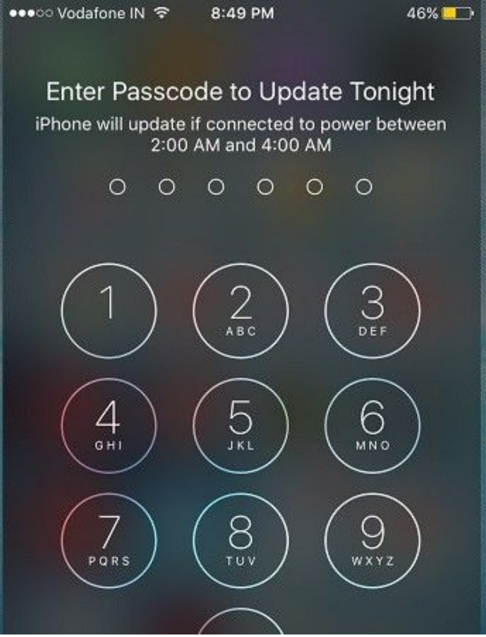

Software Update

iOS 9.3.2 is available for your iPhone and is ready to install.

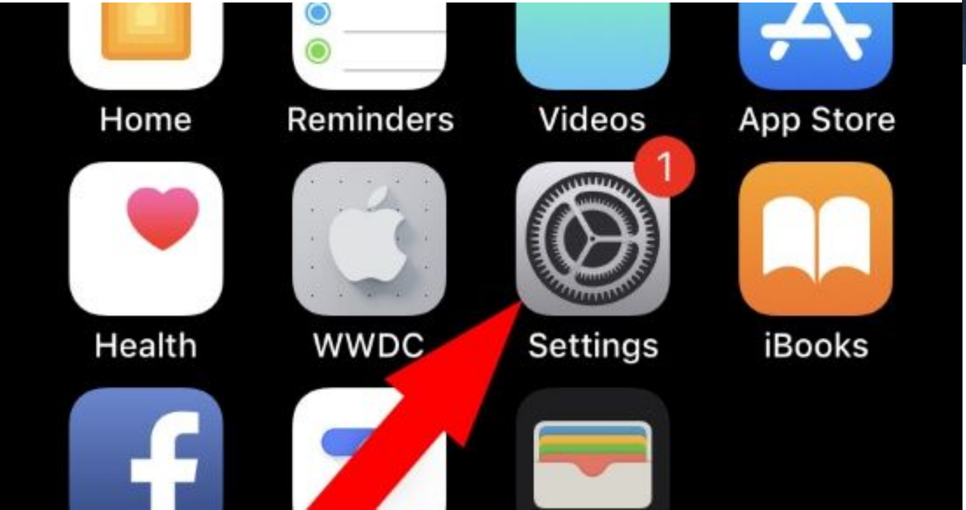
Install Now

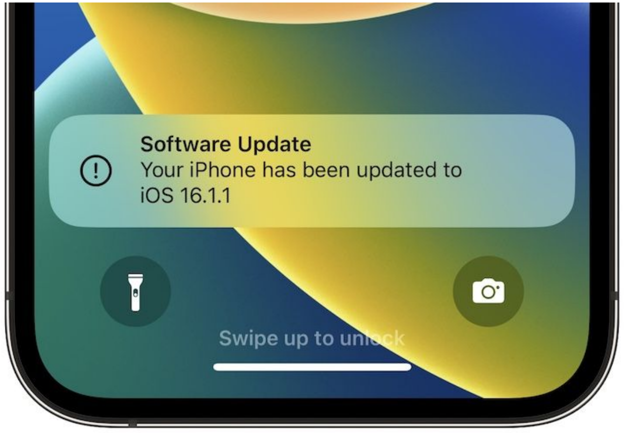
Later

Details


Enter Passcode to Update Tonight

iPhone will update if connected to power between 2:00 AM and 4:00 AM




Software Update

Your iPhone has been updated to iOS 16.1.1

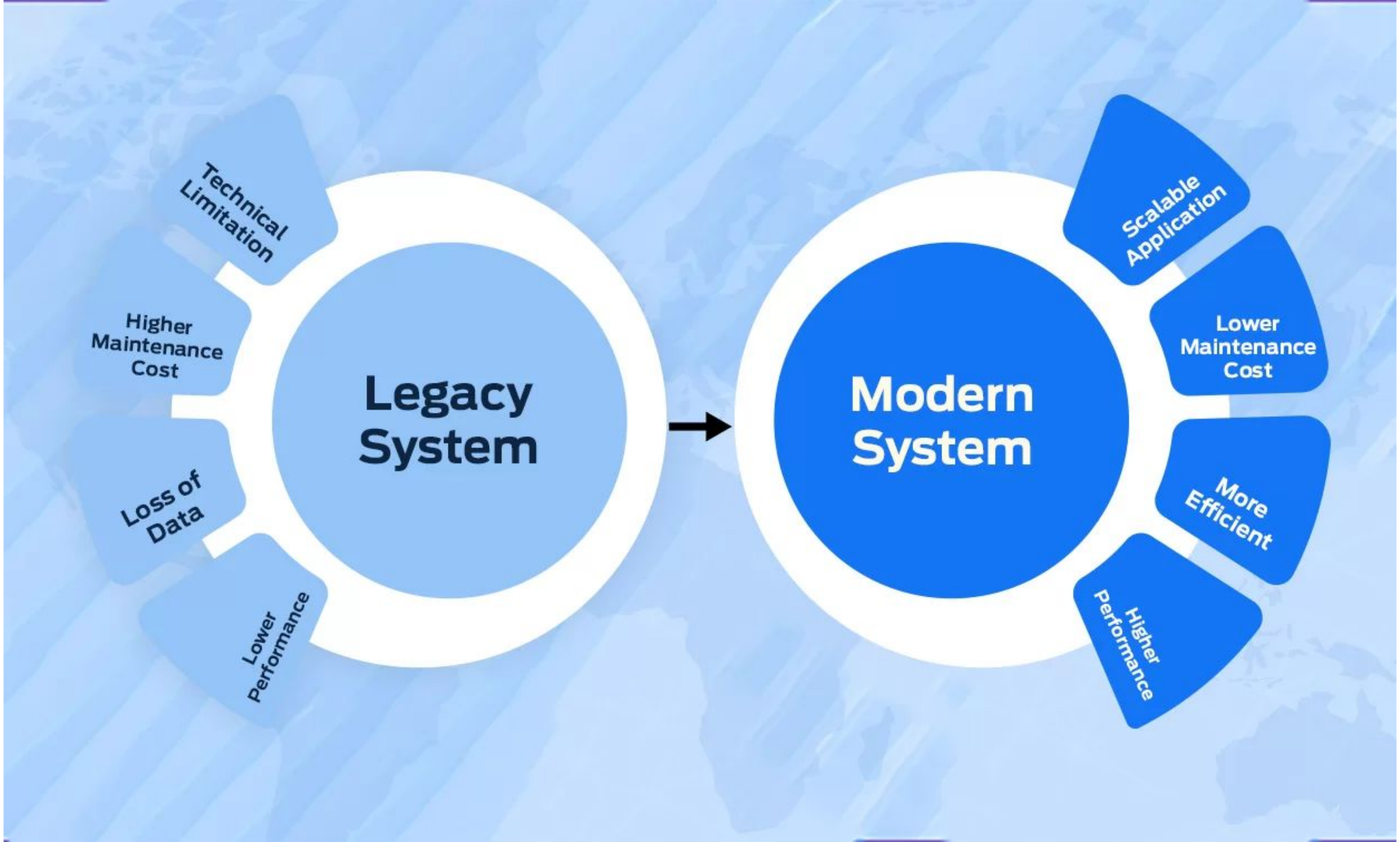
Agenda: Lesson #08 - Software Engineering - Lecture

Legacy systems

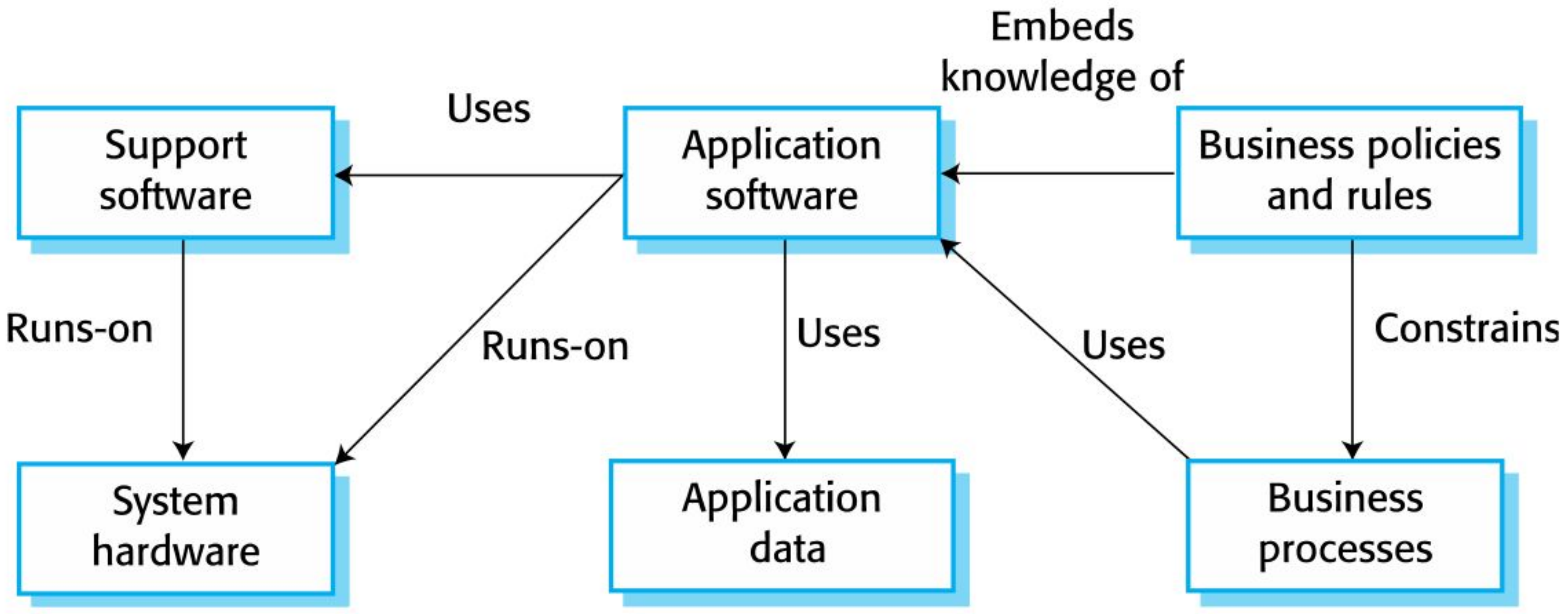
Legacy systems are older systems that rely on languages and technology that are no longer used for new systems development

Legacy software may be dependent on older hardware, such as mainframe computers and may have associated legacy processes and procedures

Legacy systems are not just software systems but are broader socio-technical systems that include hardware, software, libraries and other supporting software and business processes


Technical Limitation
Higher Maintenance Cost
Loss of Data
Lower Performance
Legacy System
Modern System
Scalable Application
Lower Maintenance Cost
More Efficient
Higher Performance

The elements of a legacy system



Legacy systems

Legacy system components

System hardware

- Legacy systems may have been written for hardware that is no longer available

Support software

- The legacy system may rely on a range of support software, which may be obsolete or unsupported

Legacy system components

Application software

The application system that provides the business services is usually made up of a number of application programs

Application data

These are data that are processed by the application system. They may be inconsistent, duplicated or held in different databases

Legacy system components

Business processes

- These are processes that are used in the business to achieve some business objective

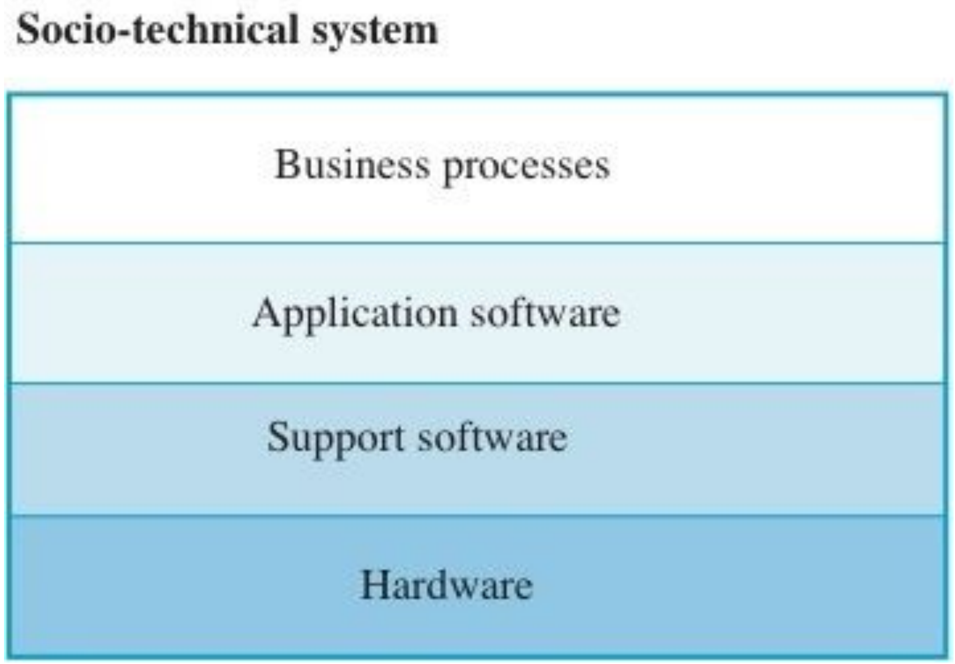
Business processes may be designed around a legacy system and constrained by the functionality that it provides

Legacy system components

Business policies and rules

These are definitions of how the business should be carried out and constraints on the business. Use of the legacy application system may be embedded in these policies and rules

Legacy system layers



Legacy system replacement

Legacy system replacement is risky and expensive so businesses continue to use these systems

System replacement is risky for a number of reasons

- Lack of complete system specification
- Tight integration of system and business processes
- Undocumented business rules embedded in the legacy system
- New software development may be late and/or over budget

Legacy system change

Legacy systems are expensive to change for a number of reasons:

- No consistent programming style
- Use of obsolete programming languages with few people available with these language skills
- Inadequate system documentation
- System structure degradation
- Program optimizations may make them hard to understand
- Data errors, duplication and inconsistency

Legacy system management

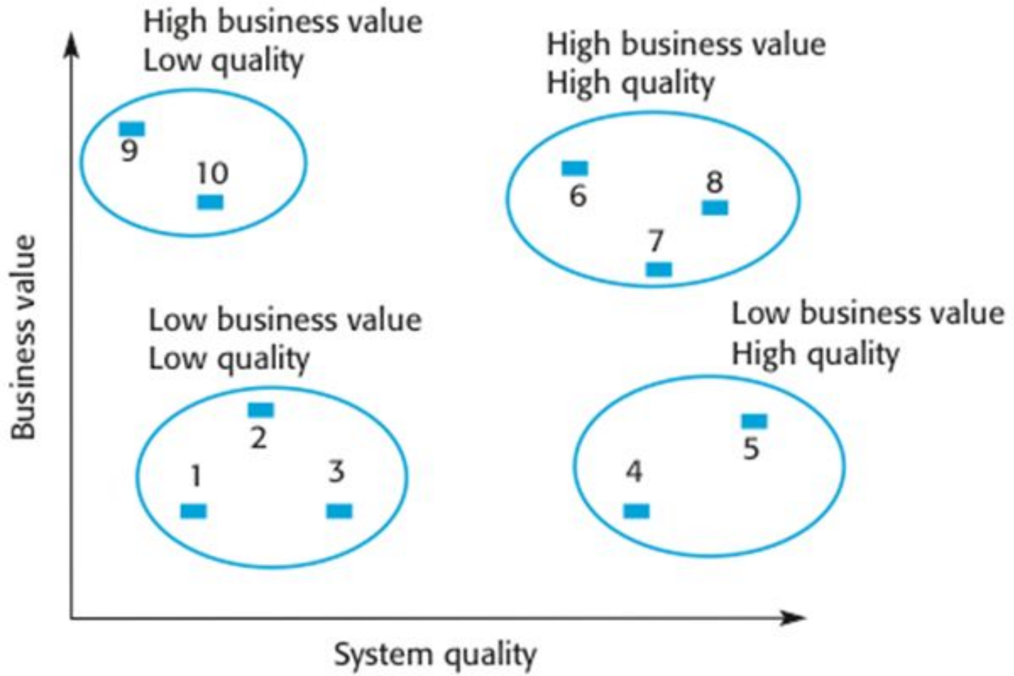
Organisations that rely on legacy systems must choose a strategy for evolving these systems

- Scrap the system completely and modify business processes so that it is no longer required;
- Continue maintaining the system;
- Transform the system by re-engineering to improve its maintainability;
- Replace the system with a new system

Legacy system management

The strategy chosen should depend on the system quality and its business value

An example of a legacy system assessment



Legacy system categories

Low quality, low business value

- These systems should be scrapped

Low-quality, high-business value

- These make an important business contribution but are expensive to maintain. Should be re-engineered or replaced if a suitable system is available

High-quality, low-business value

- Replace with COTS, scrap completely or maintain

High-quality, high business value

- Continue in operation using normal system maintenance

Business value assessment

Assessment should take different viewpoints into account

- System end-users;
- Business customers;
- Line managers;
- IT managers;
- Senior managers.

Interview different stakeholders and collate results

Issues in business value assessment

The use of the system

- If systems are only used occasionally or by a small number of people, they may have a low business value

The business processes that are supported

- A system may have a low business value if it forces the use of inefficient business processes

Issues in business value assessment

System dependability

- If a system is not dependable and the problems directly affect business customers, the system has a low business value

The system outputs

- If the business depends on system outputs, then the system has a high business value

System quality assessment

Business process assessment

- How well does the business process support the current goals of the business?

Environment assessment

- How effective is the system’s environment and how expensive is it to maintain?

Application assessment

- What is the quality of the application software system?

Business process assessment

Use a viewpoint-oriented approach and seek answers from system stakeholders

- Is there a defined process model and is it followed?
- Do different parts of the organisation use different processes for the same function?
- How has the process been adapted?
- What are the relationships with other business processes and are these necessary?
- Is the process effectively supported by the legacy application software?

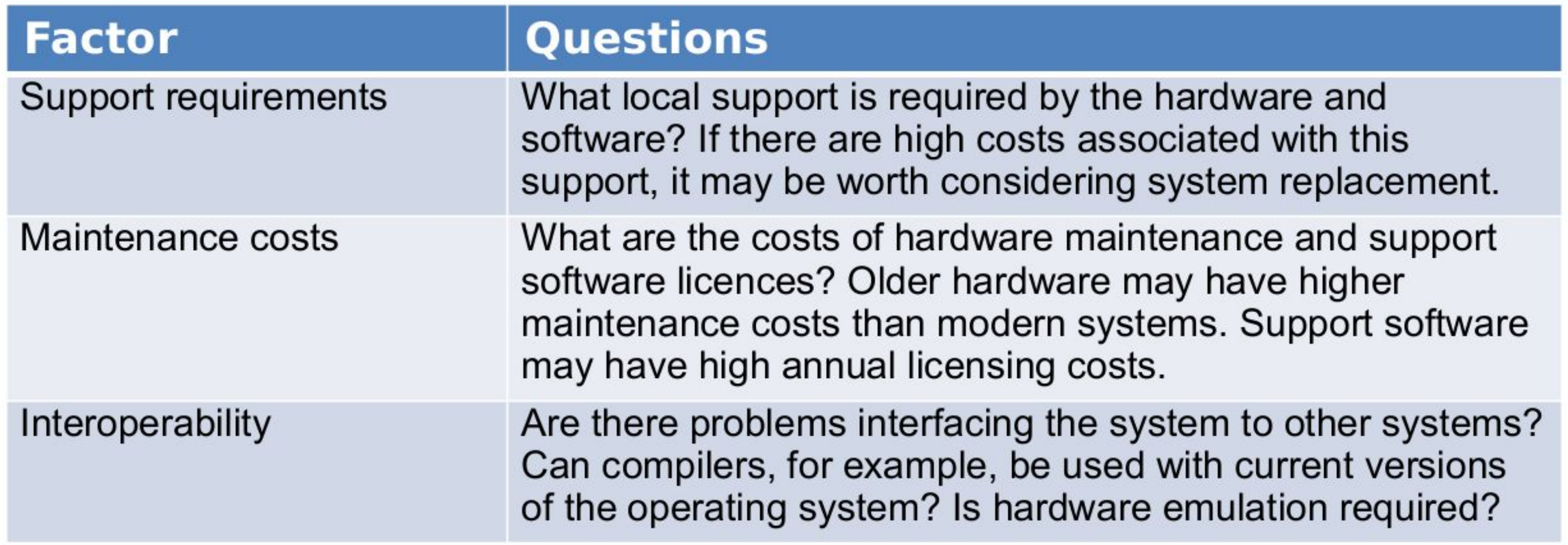
Business process assessment

Example - a travel ordering system may have a low business value because of the widespread use of web-based ordering

Factors used in environment assessment

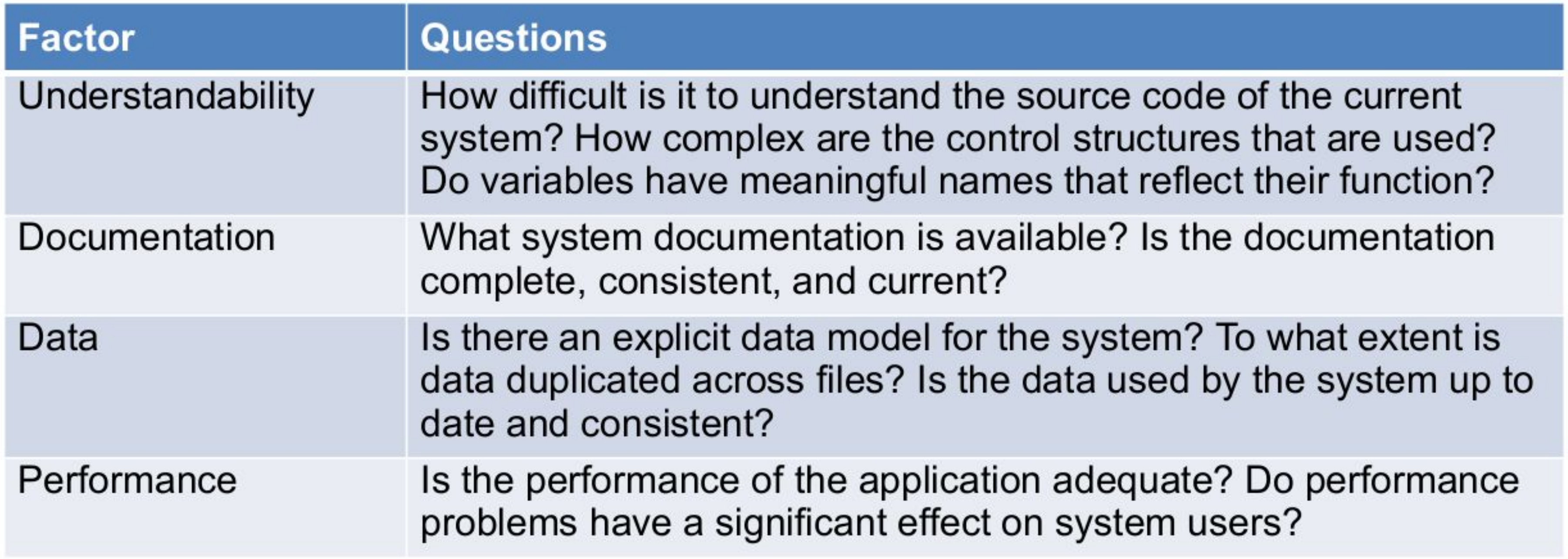
Factor	Questions
Supplier stability	Is the supplier still in existence? Is the supplier financially stable and likely to continue in existence? If the supplier is no longer in business, does someone else maintain the systems?
Failure rate	Does the hardware have a high rate of reported failures? Does the support software crash and force system restarts?
Age	How old is the hardware and software? The older the hardware and support software, the more obsolete it will be. It may still function correctly but there could be significant economic and business benefits to moving to a more modern system.
Performance	Is the performance of the system adequate? Do performance problems have a significant effect on system users?

Factors used in environment assessment



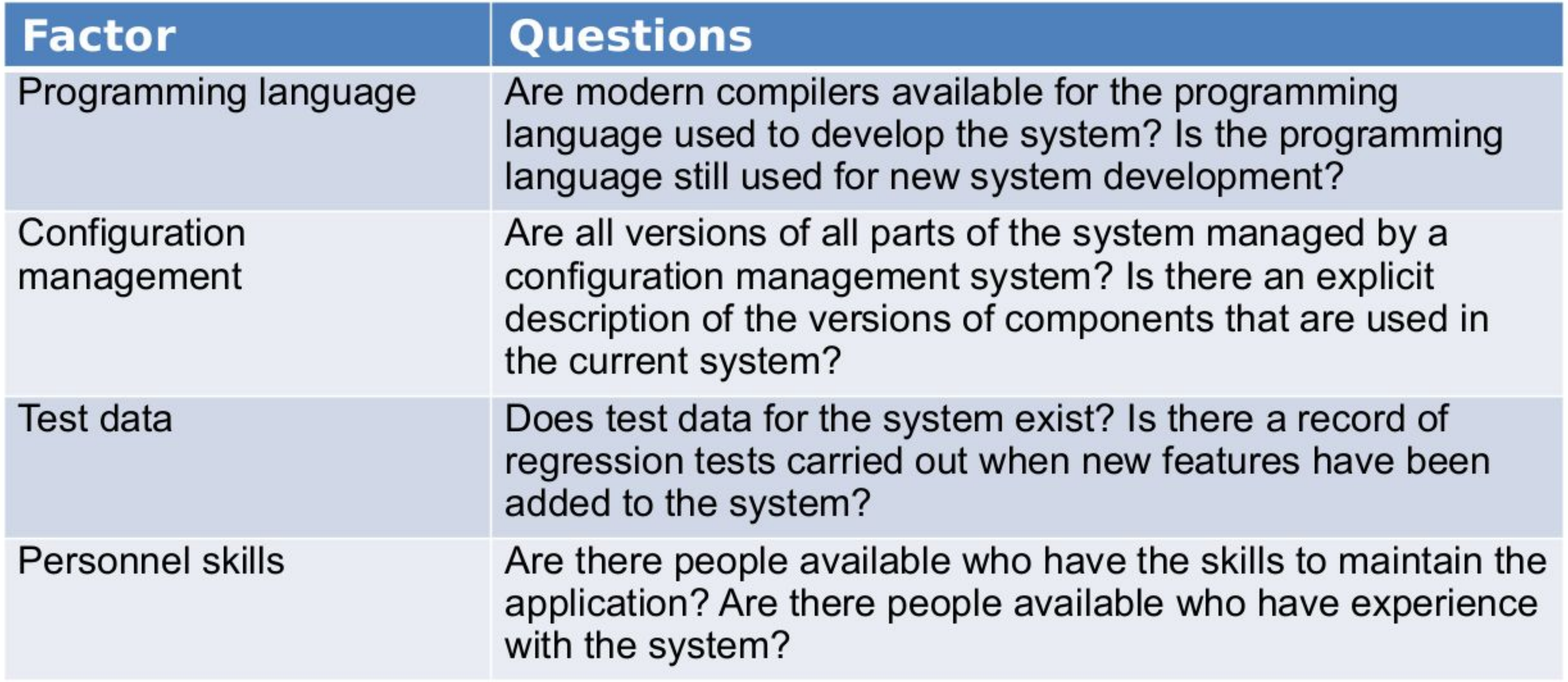
Factor	Questions
Support requirements	What local support is required by the hardware and software? If there are high costs associated with this support, it may be worth considering system replacement.
Maintenance costs	What are the costs of hardware maintenance and support software licences? Older hardware may have higher maintenance costs than modern systems. Support software may have high annual licensing costs.
Interoperability	Are there problems interfacing the system to other systems? Can compilers, for example, be used with current versions of the operating system? Is hardware emulation required?

Factors used in environment assessment



Factor	Questions
Understandability	How difficult is it to understand the source code of the current system? How complex are the control structures that are used? Do variables have meaningful names that reflect their function?
Documentation	What system documentation is available? Is the documentation complete, consistent, and current?
Data	Is there an explicit data model for the system? To what extent is data duplicated across files? Is the data used by the system up to date and consistent?
Performance	Is the performance of the application adequate? Do performance problems have a significant effect on system users?

Factors used in environment assessment



Factor	Questions
Programming language	Are modern compilers available for the programming language used to develop the system? Is the programming language still used for new system development?
Configuration management	Are all versions of all parts of the system managed by a configuration management system? Is there an explicit description of the versions of components that are used in the current system?
Test data	Does test data for the system exist? Is there a record of regression tests carried out when new features have been added to the system?
Personnel skills	Are there people available who have the skills to maintain the application? Are there people available who have experience with the system?

System measurement

You may collect quantitative data to make an assessment of the quality of the application system

- The number of system change requests; The higher this accumulated value, the lower the quality of the system

- The number of different user interfaces used by the system; The more interfaces, the more likely it is that there will be inconsistencies and redundancies in these interfaces

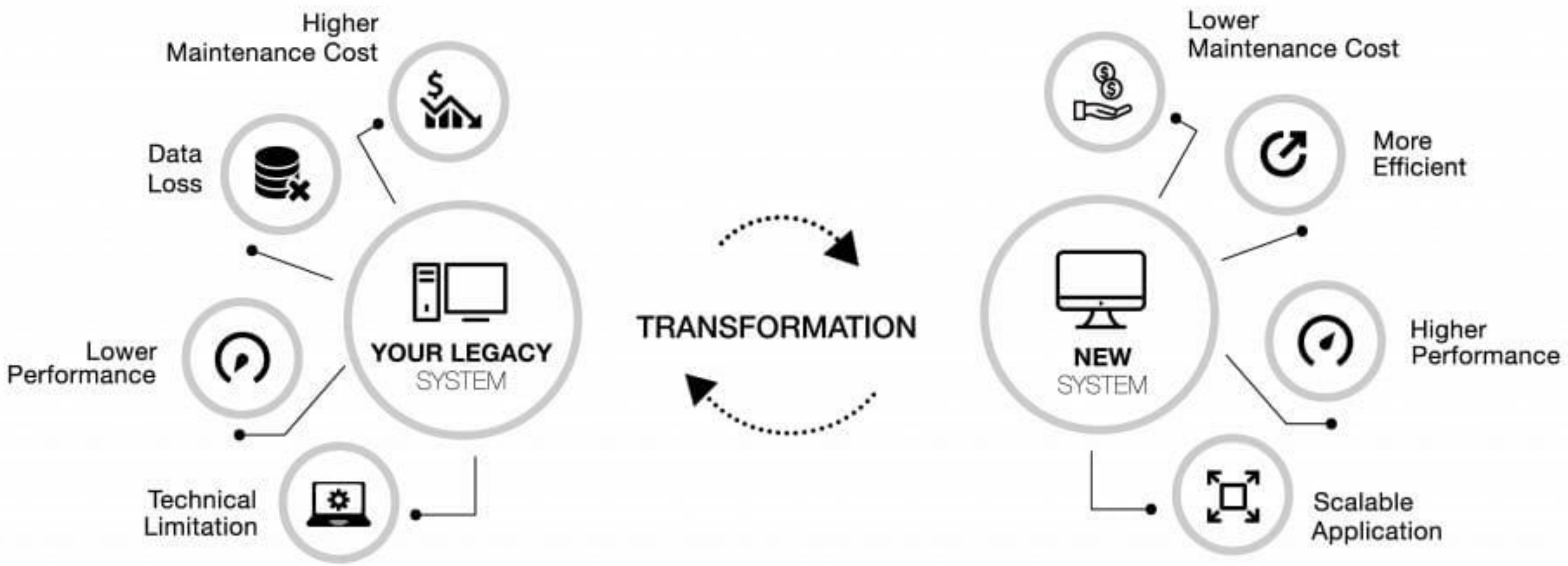
System measurement

You may collect quantitative data to make an assessment of the quality of the application system

- The volume of data used by the system. As the volume of data (number of files, size of database, etc.) processed by the system increases, so too do the inconsistencies and errors in that data

- Cleaning up old data is a very expensive and time-consuming process

Legacy systems



Agenda: Lesson #08 - Software Engineering - Lecture

Software maintenance

Modifying a program after it has been put into use

The term is mostly used for changing custom software

Generic software products are said to evolve to create new versions

Software maintenance

Software maintenance

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

Types of maintenance

Fault repairs

- Changing a system to fix bugs/vulnerabilities and correct deficiencies in the way meets its requirements

Types of maintenance

Environmental adaptation

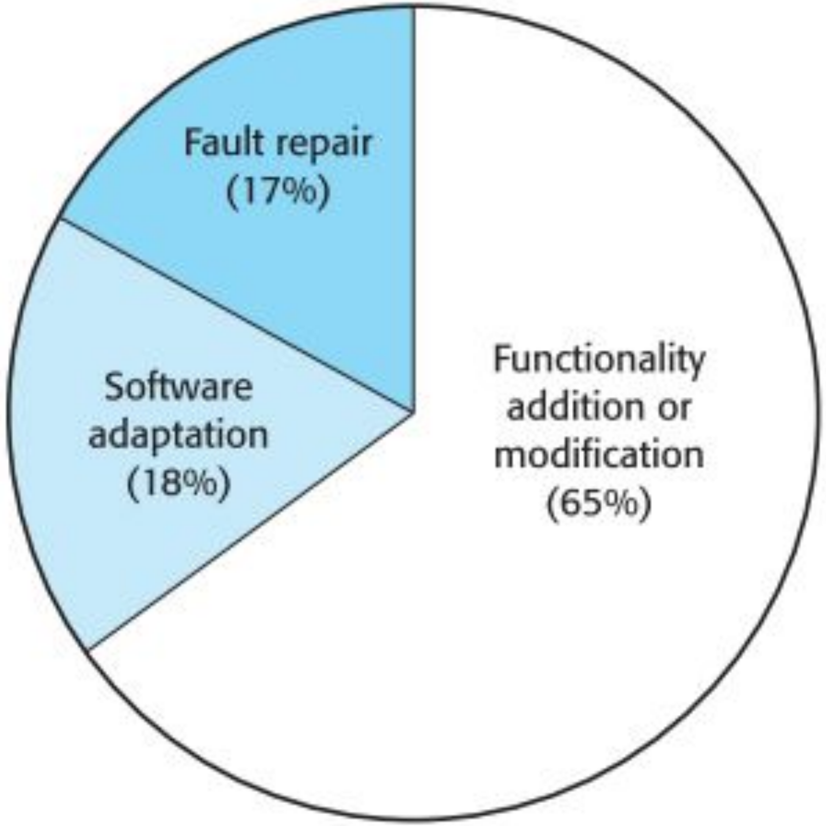
- Maintenance to adapt software to a different operating environment
- Changing a system so that it operates in a different environment (computer, OS, etc.) from its initial implementation

Types of maintenance

Functionality addition and modification

- Modifying the system to satisfy new requirements

Maintenance effort distribution



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.)

Maintenance costs

It is usually more expensive to add new features to a system during maintenance than it is to add the same features during development

- A new team has to understand the programs being maintained
- Separating maintenance and development means there is no incentive for the development team to write maintainable software

Maintenance costs

It is usually more expensive to add new features to a system during maintenance than it is to add the same features during development

- Program maintenance work is unpopular
  - Maintenance staff are often inexperienced and have limited domain knowledge

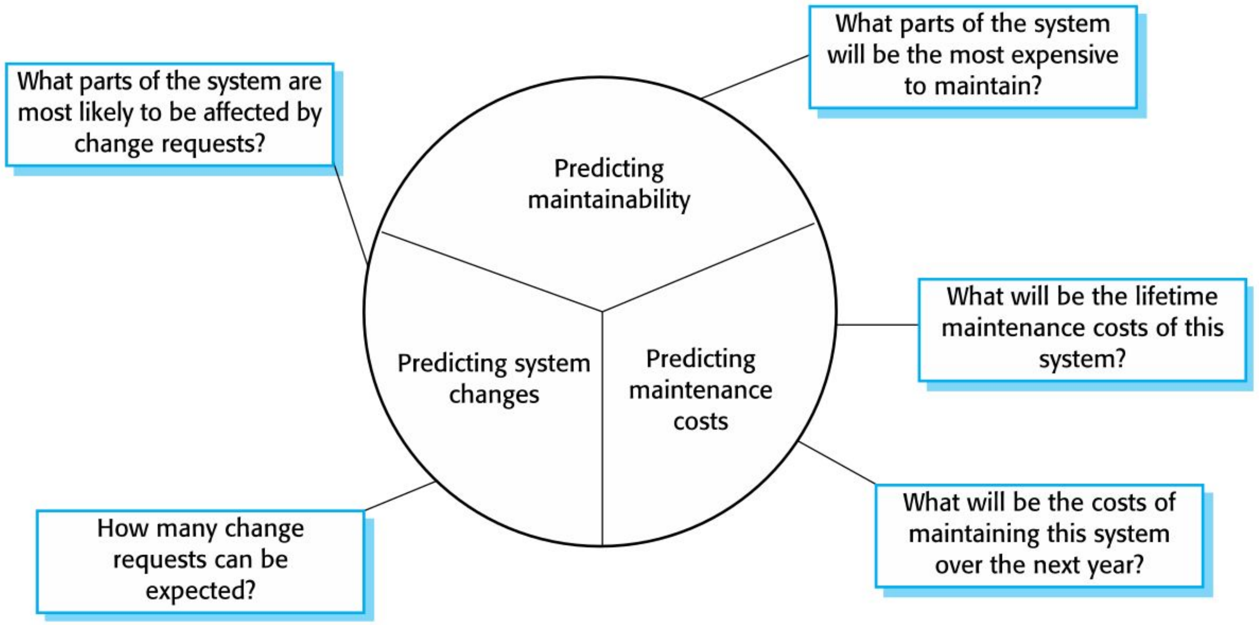
- As programs age, their structure degrades and they become harder to change

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;
- Implementing changes degrades the system and reduces its maintainability;
- 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 metrics 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

If any or all of these is increasing, this may indicate a decline in maintainability

Software reengineering

Restructuring or rewriting part or all of a legacy system without changing its functionality

Applicable where some but not all subsystems of a larger system require frequent maintenance

Reengineering involves adding effort to make them easier to maintain. The system may be re-structured and re-documented

Advantages of reengineering

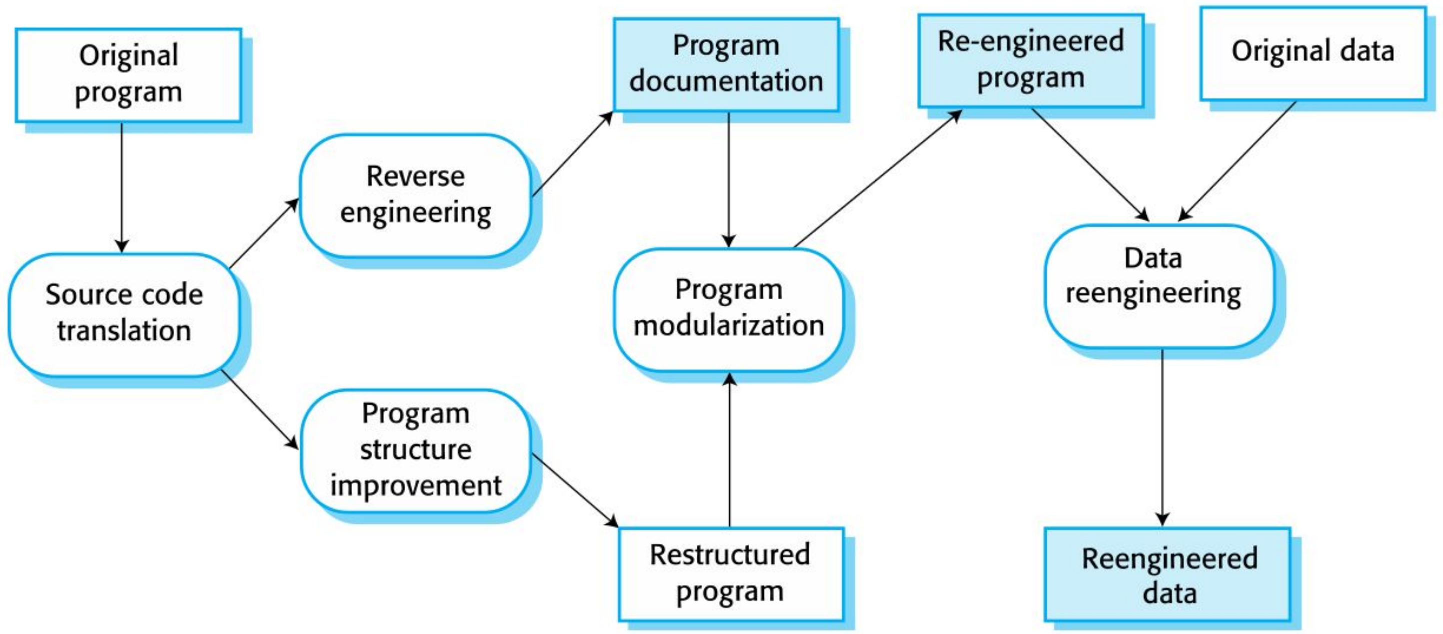
Reduced risk

- There is a high risk in new software development. There may be development problems, staffing problems and specification problems

Reduced cost

- The cost of re-engineering is often significantly less than the costs of developing new software

The reengineering process



Advantages of reengineering

Source code translation

- Convert code to a new language.

Reverse engineering

- Analyse the program to understand it;

Program structure improvement

- Restructure automatically for understandability;

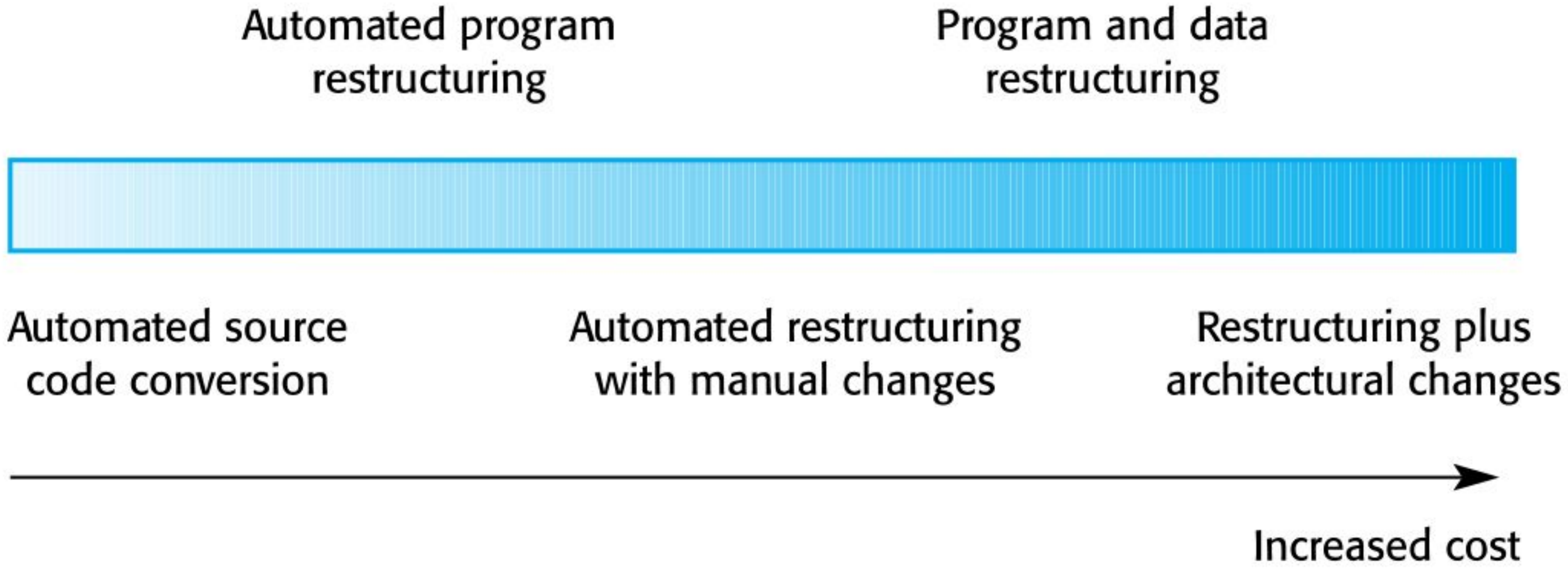
Program modularisation

- Reorganise the program structure;

Data reengineering

- Clean-up and restructure system data.

Reengineering approaches


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

Reengineering 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

Refactoring

Refactoring is the process of making improvements to a program to slow down degradation through change

You can think of refactoring as ‘preventative maintenance’ that reduces the problems of future change

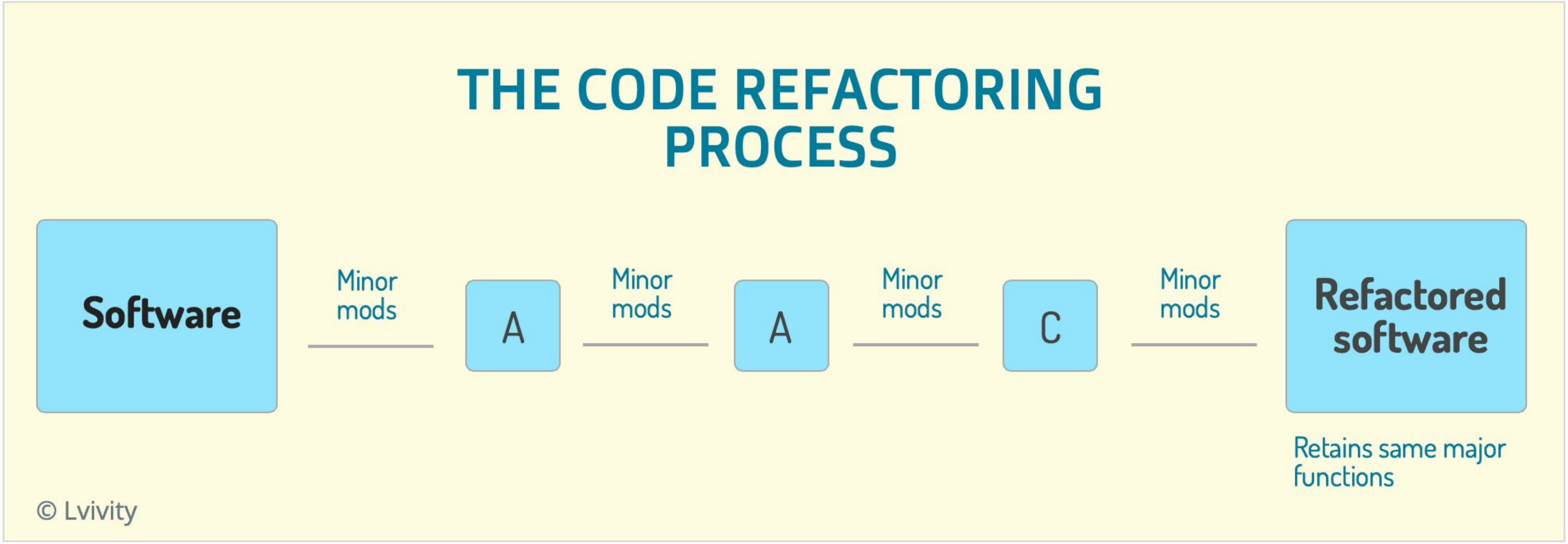
Refactoring

Refactoring involves modifying a program to improve its structure, reduce its complexity or make it easier to understand

When you refactor a program, you should not add functionality but rather concentrate on program improvement

Software maintenance

Refactoring



Refactoring and reengineering

Re-engineering takes place after a system has been maintained for some time and maintenance costs are increasing. You use automated tools to process and re-engineer a legacy system to create a new system that is more maintainable

Refactoring is a continuous process of improvement throughout the development and evolution process. It is intended to avoid the structure and code degradation that increases the costs and difficulties of maintaining a system

‘Bad smells’ in program code

Duplicate code

- The same or very similar code may be included at different places in a program. This can be removed and implemented as a single method or function that is called as required

Long methods

- If a method is too long, it should be redesigned as a number of shorter methods

‘Bad smells’ in program code

Switch (case) statements

- These often involve duplication, where the switch depends on the type of a value. The switch statements may be scattered around a program. In object-oriented languages, you can often use polymorphism to achieve the same thing

‘Bad smells’ in program code

Data clumping

- Data clumps occur when the same group of data items (fields in classes, parameters in methods) re-occur in several places in a program. These can often be replaced with an object that encapsulates all of the data

‘Bad smells’ in program code

Speculative generality

- This occurs when developers include generality in a program in case it is required in the future. This can often simply be removed

Agenda: Lesson #08 - Software Engineering - Lecture

Q & A