Elements of Software Engineering and Information Systems

Lecture 16 – Software Evolution

Prof. Ing. Lelio Campanile, PhD

Corso di Laurea Magistrale in Data Science
Dipartimento di Matematica e Fisica
Università degli Studi della Campania Luigi Vanvitelli

A.A. 2025/2026

Topics covered

Università degli Studi della Campania *Luigi Vanvitelli*

Software change

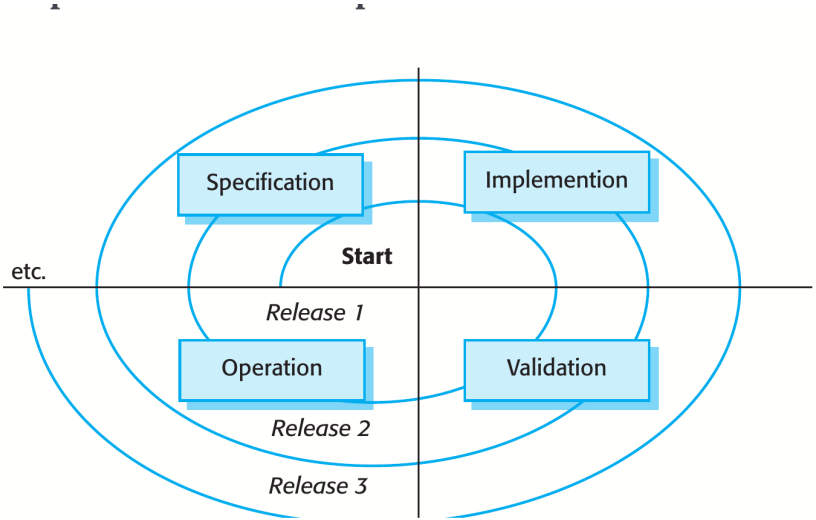
- Software change is inevitable:
  - New requirements emerge when the software is used;
  - The business environment changes;
  - Errors must be repaired;
  - New computers and equipment is added to the system;
  - The performance or reliability of the system may have to be improved.
- A key problem for all organizations is implementing and managing change to their existing software systems.

Università degli Studi della Campania *Luigi Vanvitelli*

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

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.

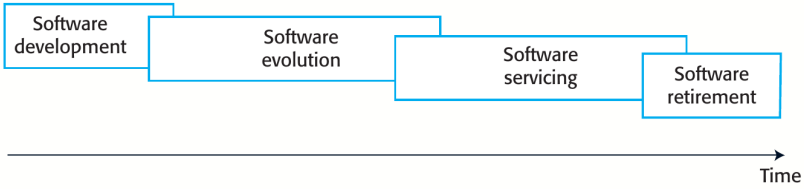
Servicing

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.

Phase-out

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

Evolution and servicing



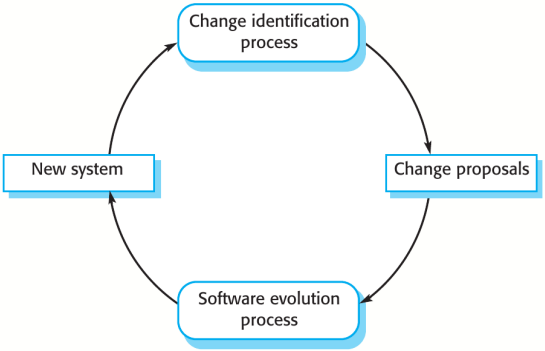
Evolution Processes

Evolution processes

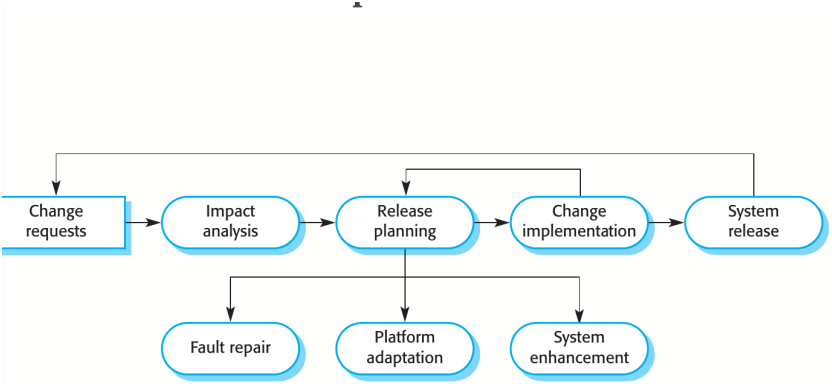
- Software evolution processes depend on:
  - The type of software being maintained;
  - The development processes used;
  - The skills and experience of the people involved.
- Proposals for change are the driver for system evolution.
  - Should be linked with components that are affected by the change, thus allowing the cost and impact of the change to be estimated.
- Change identification and evolution continues throughout the system lifetime.

Università degli Studi della Campania *Luigi Vanvitelli*

Change identification and evolution processes



The software evolution process

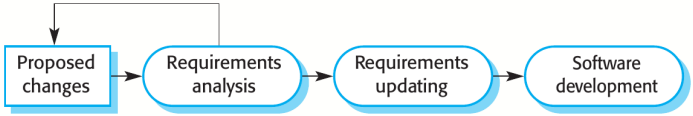


Change implementation

- 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.
- 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.

Università degli Studi della Campania *Luigi Vanvitelli*

Change implementation



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.
- 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.

Università degli Studi della Campania *Luigi Vanvitelli*

Legacy Systems

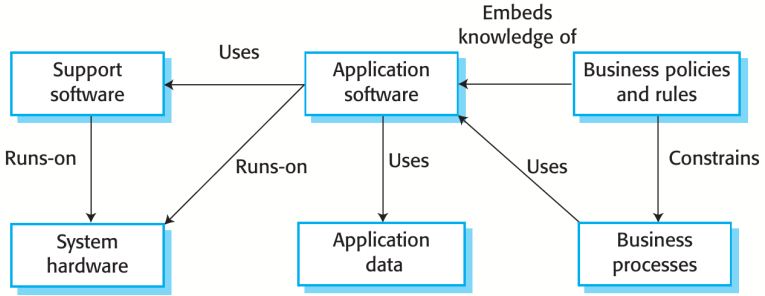
Legacy systems

- 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.

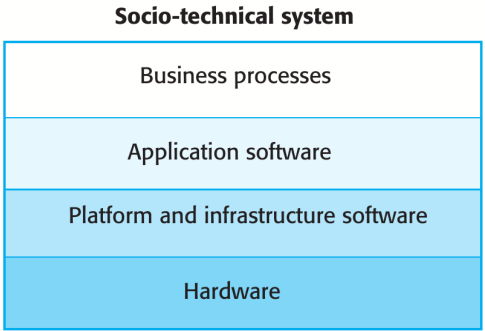
Legacy system components

- **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.
- **Application software** — The application system that provides the business services is usually made up of a number of application programs.
- **Application data** — Data that are processed by the application system. They may be inconsistent, duplicated or held in different databases.
- **Business processes** — Processes used in the business to achieve some business objective. May be designed around a legacy system and constrained by the functionality that it provides.
- **Business policies and rules** — Definitions of how the business should be carried out. Use of the legacy application system may be embedded in these policies and rules.

The elements of a legacy system



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.

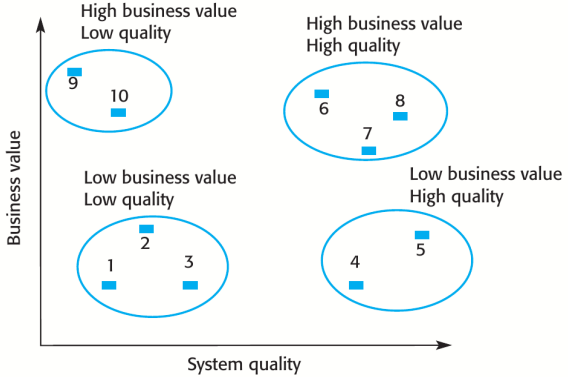
Legacy system change

- 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.
- 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.

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.

Università degli Studi della Campania *Luigi Vanvitelli*

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?
- 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?
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.
Performance	Is the performance of the system adequate? Do performance problems have a significant effect on system users?
Support requirements	What local support is required by the hardware and software?
Maintenance costs	What are the costs of hardware maintenance and support software licences?
Interoperability	Are there problems interfacing the system to other systems?

Factors used in application assessment

Factor	Questions
Understandability	How difficult is it to understand the source code of the current system?
Documentation	What system documentation is available? Is it complete, consistent, and current?
Data	Is there an explicit data model for the system? To what extent is data duplicated across files?
Performance	Is the performance of the application adequate?
Programming language	Are modern compilers available for the programming language used to develop the system?
Configuration management	Are all versions of all parts of the system managed by a configuration management system?
Test data	Does test data for the system exist? Is there a record of regression tests?
Personnel skills	Are there people available who have the skills to maintain the application?

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 there will be inconsistencies and redundancies.
  - The volume of data used by the system — as the volume of data increases, so too do the inconsistencies and errors in that data.
  - Cleaning up old data is a very expensive and time-consuming process.

Software Maintenance

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.
- 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 (Corrective maintenance)
Changing a system to fix bugs/vulnerabilities and correct deficiencies in the way it meets its requirements.

Environmental adaptation (Adaptive maintenance)
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.

Functionality addition and modification (Perfective maintenance)
Modifying the system to satisfy new requirements.

Università degli Studi della Campania *Luigi Vanvitelli*

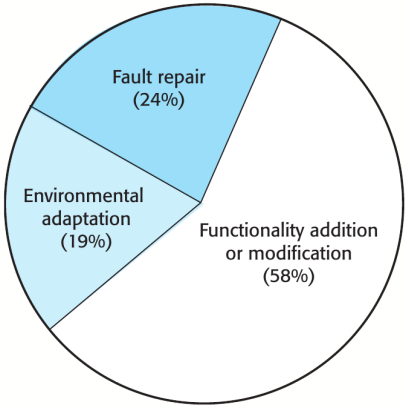
Maintenance costs

- 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 (cont.)

- 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;
  - 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 effort distribution



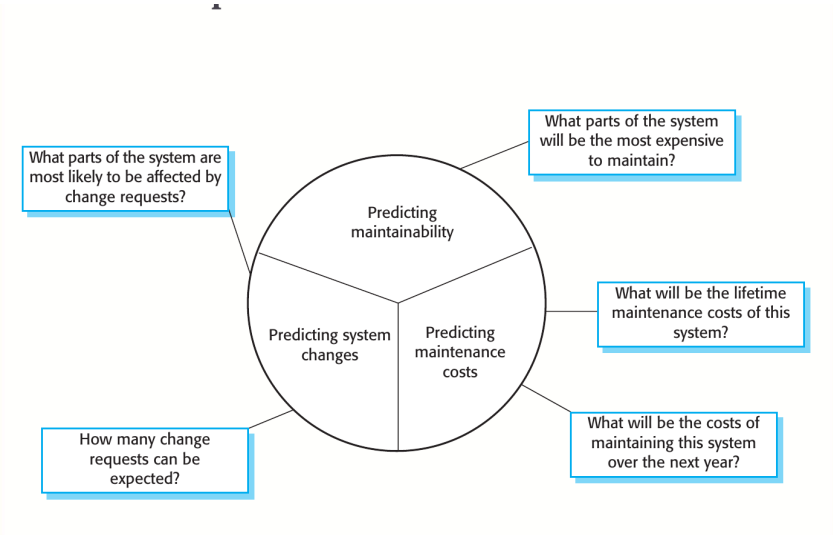
Maintenance prediction

- 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.

Change prediction

- Predicting the number of changes requires an 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.

Maintenance prediction



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 sub-systems 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

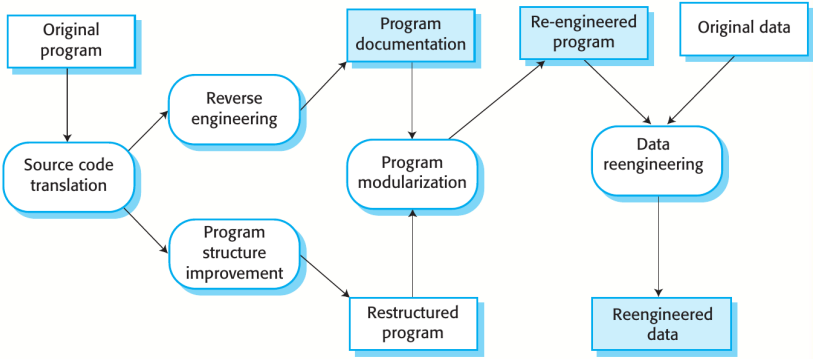
Reduced risk

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



Reengineering process activities

Source code translation
: Convert code to a new language.

Reverse engineering
: Analyse the program to understand it.

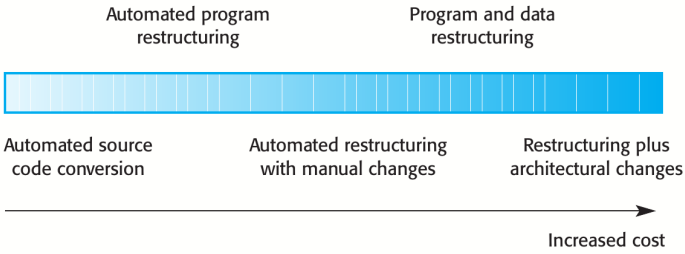
Program structure improvement
: Restructure automatically for understandability.

Program modularisation
: Reorganise the program structure.

Data reengineering
: Clean-up and restructure system data.

Università degli Studi della Campania *Luigi Vanvitelli*

Reengineering approaches



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 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.

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.

Switch (case) statements

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

Data clumping

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

Speculative generality

This occurs when developers include generality in a program in case it is required

Key Points

Key points

- Software development and evolution can be thought of as an integrated, iterative process that can be represented using a spiral model.
- For custom systems, the costs of software maintenance usually exceed the software development costs.
- The process of software evolution is driven by requests for changes and includes change impact analysis, release planning and change implementation.
- Legacy systems are older software systems, developed using obsolete software and hardware technologies, that remain useful for a business.

Key points (cont.)

- It is often cheaper and less risky to maintain a legacy system than to develop a replacement system using modern technology.
- The business value of a legacy system and the quality of the application should be assessed to help decide if a system should be replaced, transformed or maintained.
- There are 3 types of software maintenance, namely bug fixing, modifying software to work in a new environment, and implementing new or changed requirements.

Key points (cont.)

- Software re-engineering is concerned with re-structuring and re-documenting software to make it easier to understand and change.
- Refactoring, making program changes that preserve functionality, is a form of preventative maintenance.

Grazie per l'attenzione

Prof. Ing. Lelio Campanile, PhD

Professore a contratto

Dipartimento di Matematica e Fisica

Elements of Software Engineering and Information Systems

Corso di Laurea Magistrale in Data Science

Università degli Studi della Campania Luigi Vanvitelli