*Legacy systems* : old computer-based systems still in use by organizations

Many of them still business critical

Incorporate many changes made over the years

Many people have been involved in these changes

Replacing legacy systems with new systems is risky, yet keeping them means new changes become more and more expensive

### ****Risks of Replacing a Legacy System:****

1. **Incomplete documentation**: It’s hard to fully understand how the old system works because the documentation is often missing or outdated.
2. **Expensive to change**: The business has adjusted to the old system, and changing processes to fit a new system can be expensive.
3. **Hidden business rules**: The old system may have important rules built in that aren’t documented, and the new system might break them.
4. **Delays and extra costs**: The new system may take longer to develop, cost more, and not work as expected.

### ****Why Changing Legacy Systems is Expensive:****

1. **Inconsistent code** – Different teams wrote different parts of the system, making it messy and hard to update.
2. **Lack of skilled developers** – Old systems use outdated programming languages, and few people know how to work with them.
3. **Poor or missing documentation** – Often, the only way to understand the system is through the source code, which may not even be available.
4. **Complex updates** – The system has been modified over the years without a clear plan, making changes difficult.
5. **Data issues** – The system's data may be outdated, redundant, or hard to work with, adding to the complexity.

Legacy systems involve more than software (they are computer-based systems). Typical logical parts of a legacy system are:

System hardware

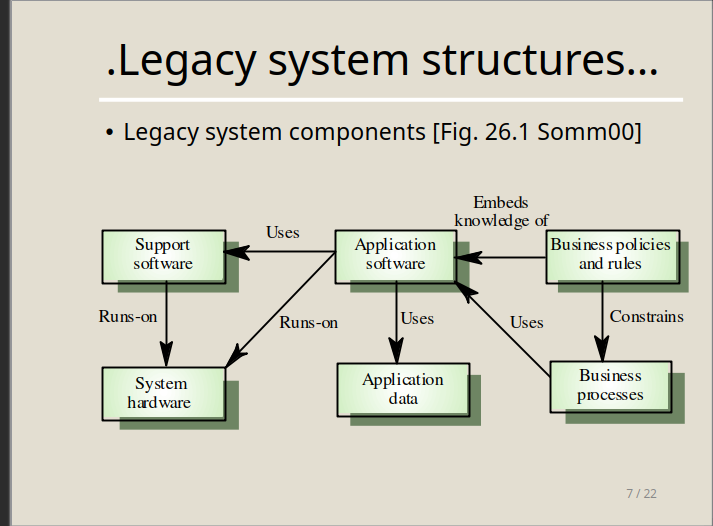
Support software

Application software (legacy software systems)

Application data

Business processes

Business policies and rules



This diagram represents the structure of a **legacy system**, which is an old but still functional software system used in an organization. Let’s break down the components in simple terms:

1. **System Hardware** 🖥️
   * The physical computers, servers, or devices on which the software runs.
2. **Support Software** 🛠️
   * Additional software (like operating systems, databases, or utilities) that helps the main application software function.
   * It runs on **System Hardware** and is used by **Application Software**.
3. **Application Software** 📦
   * The main program that performs the system’s key tasks (e.g., payroll system, inventory management).
   * Runs on **System Hardware** and uses **Support Software** and **Application Data**.
4. **Application Data** 📊
   * The information that the application software processes (e.g., employee records, sales data).
5. **Business Policies & Rules** 📜
   * Guidelines and rules that the business follows (e.g., discount policies, tax rules).
   * These rules are embedded within the **Application Software**.
6. **Business Processes** 🔄
   * The actual workflows and tasks performed in the business (e.g., generating reports, approving orders).
   * Constrained by **Business Policies & Rules**.

### How They Connect:

✅ **Application Software** runs on **System Hardware** and uses both **Support Software** and **Application Data**.  
✅ It follows **Business Policies & Rules**, which define how it should work.  
✅ These rules affect **Business Processes**, guiding how the business operates.

In short, this diagram shows how different parts of a legacy system depend on each other to keep a business running! 🚀

Legacy systems design is typically function oriented. Two main classes:

*Batch processing systems*: both input and output is provided in “batches,” e.g., a payroll system

*Transaction processing systems*: input & output related to a database transaction, e.g., a flight reservation system

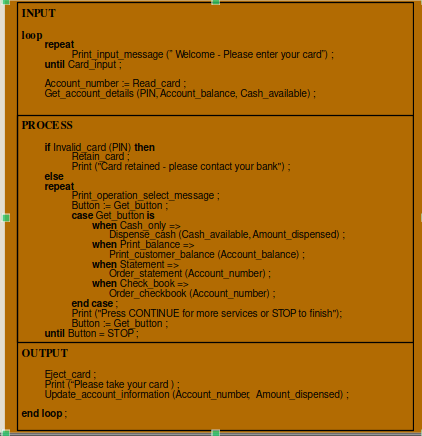
Both batch processing and transaction processing systems usually follow an *IPO model*:

Input: inputs are collected from one or more sources

Processing: some computations are performed on inputs

Output: results are provided either in batches or as single-transaction outputs

All IPO components may further be organized according the IPO model



**Strategic approaches for dealing with legacy systems:**

**Scrap the system completely**

When business practices have changed and no longer depend significantly on the system (they may be supported by new COTS)

**Continue to maintain the system**

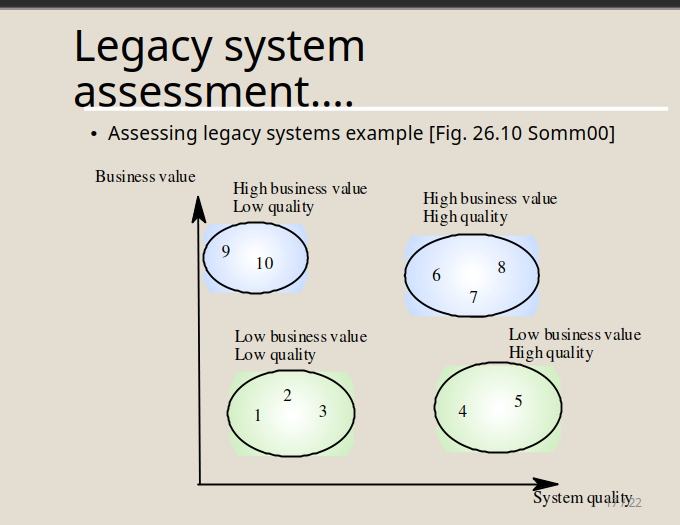
The system works well, is fairly stable, and users do not request many changes

**Transform the system to improve maintainability**

When system quality was affected negatively by changes, yet changes are still required

**Replace the system with a new one**

When obsolete hardware precludes further operation or the new system can be built at reasonable cost



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

Assessment of legacy systems includes:

Business value assessment (subjective). Viewpoints:

End-users: look at system’s functionality and performance

Customers: look at the quality of services provided

Business managers: assess the usefulness of the system in terms of business support

IT managers: are concerned with the availability of technical support for the system

Senior managers: interested in system’s contribution to the business goals

System quality assessment (next)

Key points

A legacy system is an old system that still provides essential business services

Legacy systems are not just application software but also include business processes, support software and hardware

Most legacy systems are made up of several different programs and shared data

A function-oriented approach has been used in the design of most legacy systems

The structure of legacy business systems normally follows an input-process-output model

The business value of a system and its quality should be used to choose an evolution strategy

The business value reflects the system’s effectiveness in supporting business goals

System quality depends on business processes, the system’s environment and the application software