22

Legacy Information System

Shah Nawaz,
Lecturer, Department of Software Engineering,
Lahore Garrison University, shahnawaz@lgu.edu.pk

Legacy Information System

Legacy Information System

General Idea

- word “legacy” is defined as “A sum of money, or a specified article, given to another by will; anything handed down by an ancestor or predecessor.”

- In computing, a legacy system is an old method, technology, computer system, or application program, "of, relating to, or being a previous or outdated computer system," yet still in use.

Legacy Information System

General Idea

features of a legacy system:

- large with millions of lines of code;
- geriatric, often more than 10 years old;
- written in obsolete programming languages;
- lack of consistent documentation;
- poor management of data, often based on flat-file structures;
- degraded structure following years of modifications;
- very difficult, if not impossible, to expand;
- runs on old processors which are generally much slower; and
- the support team is totally reliant on the expertise and knowledge of expert individuals.

Legacy Information System

Solutions for Legacy Software

- Due to the lack of available skills and flexibility of the system, there is a strong urge to get rid of the legacy software system as soon as possible and start with something modern.
- There are several categories of solutions for legacy information system (LIS) or simply legacy system. These solutions generally fall into six categories as follows:

Legacy Information System

Freeze.

- The organization decides to not carry out further work on an LIS.

Outsource.

- An organization may decide that supporting legacy (or any) software is not its core business. The organization may outsource it to a specialist organization offering this service.

Carry on maintenance.

- Despite all the problems associated with supporting a software system, the organization decides to carry on maintenance for another period.

Legacy Information System

Discard and redevelop.

- The organization throws all the software away and redevelops the application once again from scratch, using a new hardware platform and modern architecture tools and database. This approach removes the legacy problem in one stroke, but also removes a useful source of information for defining the requirements of the new system.

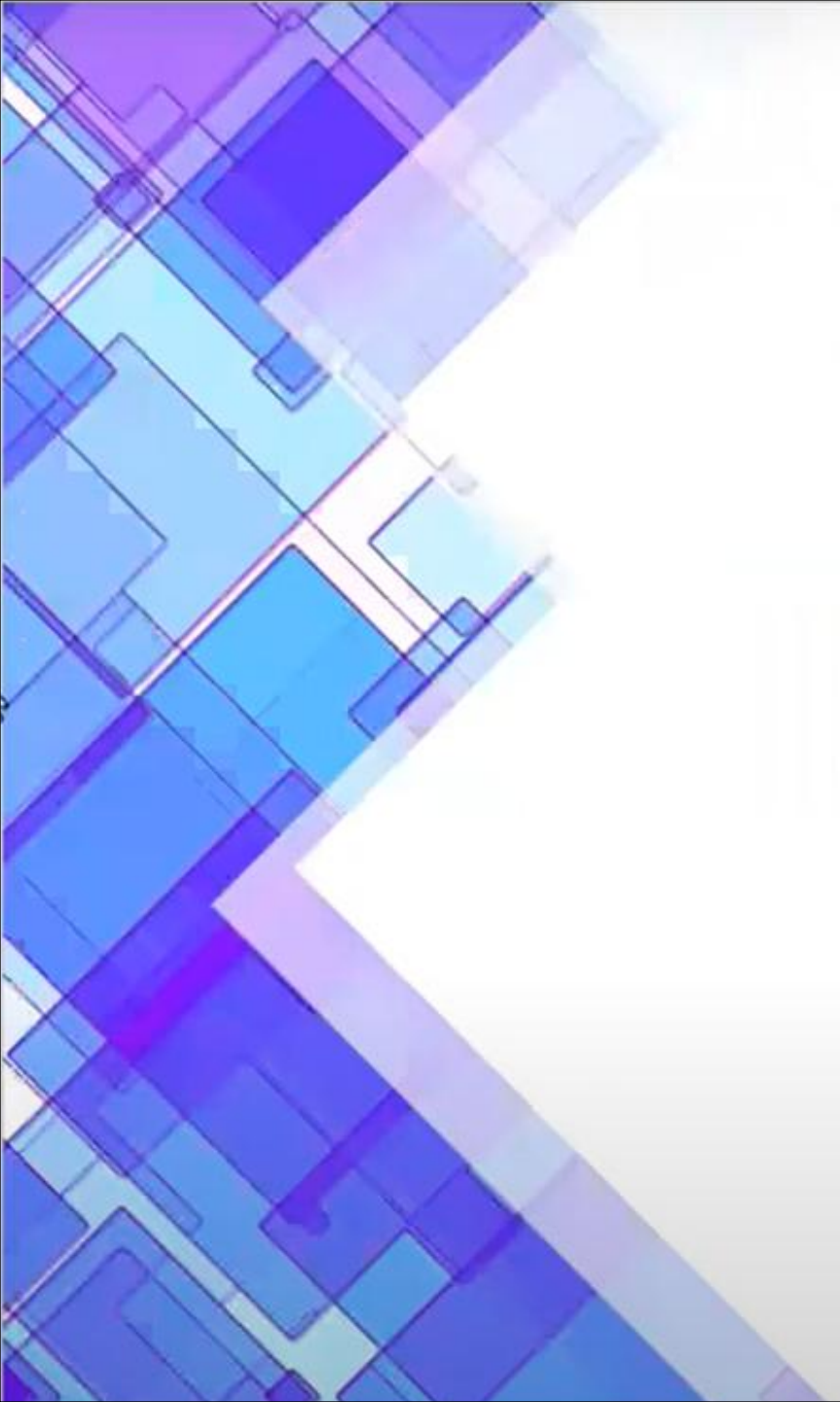
Wrap.

- It is a black-box-based modernization technique: surround the LIS with a new layer of software to hide the complexity of the existing interfaces, applications, and data, with the new interfaces. Wrapping gives existing components a modern and improved appearance.

Legacy Information System

Migrate.

- By means of migration a legacy system is ported to a modern platform, while retaining most of the system's functionality and introducing minimal disturbance in the business environment. Migration avoids the expensive and long process that generally characterize new development. Rather, reuse of portions of the LIS, namely, implementation, design, specification, and requirements, is maximized. In addition, the target system runs in a different computing environment.



Wrapping

Wrapping

- In 1988, Dietrich first introduced the concept of a “wrapper” at IBM.
- Wrapping is a straightforward way to modify and enhance a legacy component.
- A wrapper does not directly modify the source code, but, instead, it indirectly modifies and changes the software functionality of the legacy component. Wrapping means encapsulating the legacy component with a new software layer that provides a new interface and hides the complexity of the old component.
- Wrapping is a “black-box” reengineering activity because the interface of the legacy system is analyzed but not its internal details.

Wrapping

- In a software context, the term “wrapper” refers to programs or codes that literally wrap around other program components. Several different wrapper functions can be distinguished.
- They are often used for ensuring compatibility or interoperability between different software structures. Alternatively, they can be used for visual reasons, as is the case with HTML or CSS wrappers. Wrappers can be individual software components, independent software products, software architectures, classes in object-oriented programming, or frameworks.

Types of Wrapping

Wrappers are classified into four categories depending upon what is being wrapped.

Database wrappers.

Database wrappers serve as entry points to existing databases. Those wrappers enable clients implemented in today's object oriented languages to access legacy databases. Database wrappers can be further classified into

- *forward wrappers* and
- *backward wrappers.*

The forward wrappers' approach shows the process of adding a new component to a legacy system. The new component is developed with modern practices, whereas the legacy database and the legacy code are not modified. Those are called *forward wrappers* because they emulate new technology.

In backward wrappers' approach, data are migrated first following the *Database first* approach.

Types of Wrapping

System service wrappers. This kind of wrappers support customized access to commonly used system services, namely, routing, sorting, and printing. A client program may access those services without knowing their interfaces.

Application wrappers. This kind of wrappers encapsulate online transactions or batch processes. These wrappers enable new clients to include legacy components as objects and invoke those objects to produce reports or update files.

Function wrappers. This kind of wrappers provide an interface to call functions in a wrapped entity. In thismechanism, only certain parts of a program—and not the full program—are invoked from the client applications. Therefore, limited access is provided by function wrappers.

Levels of Encapsulation

Levels of Encapsulation

- Legacy databases can be accessed at that level at which the database system operates. If one is dealing with file systems one will get records.
- There are in effect five levels at which one can access legacy software applications:
- – at the process level,
- – at the transaction level,
- – at the program level,
- – at the module level, and
- – at the procedural level.

Levels of Encapsulation

- **At the process level**
- a job is started after having allocated and filled its input files via file transfers from the client application. To this end a job control or command procedure is generated by the wrapper. When the job is finished, the output files are taken over by the wrapper and forwarded back to the client. This is the simplest form of encapsulation.
- **At the transaction level**
- requests from the client application are converted into transactions of a teleprocessing monitor. However, instead of receiving panels from a terminal the transaction processing programs are fed a data stream from the client application. The output messages produced by the transaction programs have to be intercepted by the wrapper and redirected back to the client application rather than going to an end terminal. This form of wrapping has become relatively simple since modern transaction processors have solved the problem of host to client communications and taken care of exception handling as well as the tasks of restart and recovery, commit and rollback.

Levels of Encapsulation

- **At the program level**
- batch programs are called via an application program interface. The inputs which drive the program logic are simulated by a wrapper which substitutes them with data streams from the client application. Program outputs are also captured and returned to the client. These could be files, displays or reports.
- **At the module level**
- existing modules are loaded and executed using their standard linkage interface. The only difference to the normal call is that the parameters are passed by value rather than by reference, i.e., the parameter values are stored in the address space of the wrapper after having been received from the client. Afterwards, the output parameter values are forwarded back to the client. At the procedure level an internal procedure is invoked as if it were a separately compiled module. This entails building a parameter interface and possibly setting global values prior to calling the procedure. This can become the most challenging form of wrapping.