Legacy Systems - Definition and Wrappers

Idaho State University | Computer Science

Isaac Griffith

CS 4423 and CS 5523
Department of Computer Science
Idaho State University

ROAR

Idaho State University Computer Science

Outcomes

After today's lecture you will:

- Be able to describe the basic characteristics of a Legacy System
- Understand the primary methods of dealing with legacy systems
- Be able to describe the different approaches in wrapping a legacy system

ROAR

Legacy Systems

CS 4423/5523

ROAR

Some Definitions

- **Legacy** - A sum of money, or a specified article, given to another by will; anything handed down by an ancestor or predecessor. (OED)
- Legacy System (Brodie and Stonebraker):
  - any information system that significantly resists modification and evolution to meet new and constantly changing business requirements.
- Legacy System (Bennett):
  - Legacy software systems are large software systems that we don’t know how to cope with but that are vital to our organization.

Legacy System Features

- 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 processor.

Legacy System Solutions

- There are several categories of solutions for legacy information systems (LIS)
- These fall into six categories:
  - Freeze
  - Outsource
  - Carry on maintenance
  - Discard and redevelop
  - Wrap
  - Migrate

Wrapping

CS 4423/5523

ROAR

Wrapping

- In 1988, Dietrich et al. first introduced the concept of a “wrapper” at IBM.
- Wrapping means encapsulating the legacy component with a new software layer that provides a new interface and hides the complexity of the old component.
- The encapsulation layer can communicate with the legacy component through sockets, remote procedure calls (RPCs), or predefined application program interfaces (API).
- The wrapped component is viewed similar to a remote server; it provides some service required by a client that does not know the implementation details of the server.

Wrapping

- By means of a message passing mechanism, a wrapper connects to the clients.
- On the input front, the wrapper accepts requests, restructures them, and invokes the target object with the restructured arguments.
- On the output front, the wrapper captures outputs from the wrapped entity, restructures the outputs, and pushes them to the requesting entity.
- However, this technique does not solve the problems with legacy systems.

Types of Wrapping

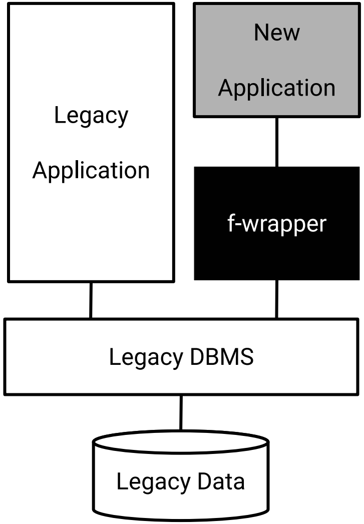
- Orfali et al. classified wrappers into four categories:
  - Database wrappers.
  - System service wrappers.
  - Application wrappers.
  - Function wrappers.

Database Wrappers

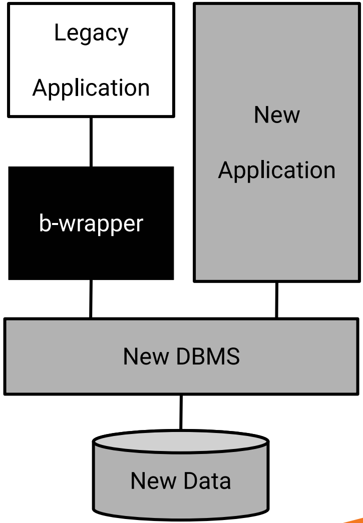
- Database wrappers can be further classified into forward wrappers (f-wrappers)and backward wrappers (b-wrappers)
- The forward wrappers-approach, depicted in Figure 5.1, shows the process of adding a new component to a legacy system.
- Therefore, by means of translation service involving both legacy data and queries, the wrapper integrates the new component with the legacy system.
- The backward wrappers-approach has been depicted in Figure 5.2
- In this approach, data are migrated first then new components are developed that use the new database; the legacy components access the new data via wrappers.

Database Wrappers

Forward Wrapper



Backward Wrapper



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

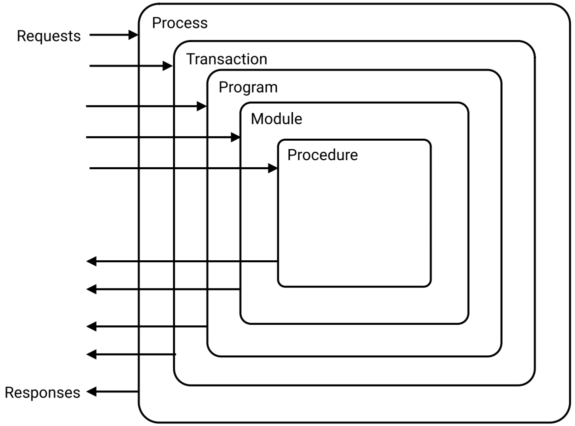
- This kind of wrappers encapsulate online transactions or batch processes.
- These wrappers enable new clients to include legacy components as objects and invoke those objects to produce reports or update files.

Function Wrappers

- This kind of wrappers provide an interface to call functions in a wrapped entity.
- In this mechanism, 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

- Sneed classified **five** levels of granularity:
  - **procedures** are at the lowest level
  - **processes** are at the highest level



Process Level

- A job is started on the server which accepts input data, accesses databases, and produces output data.
- The input data and output data are contained in files.
- The input files are created by the client program and are transferred to the server by the wrapper by means of a standard file transfer utility.
- Upon the completion of the job, the wrapper takes over the output files to be forwarded to the client.

Transaction Level

- On the server, a virtual terminal is created that sends the terminal input map and receives the terminal output map.
- The wrapper is a program which simulates the user terminal.
- This type transaction level wrapping has become simple because modern transaction processors:
    1. take care of host-to-client communication.
    2. perform restart and recovery task, exception handling, rollback, and commit.

Program Level

- Via APIs, batch programs are called in a wrapper.
- The wrapper substitutes program inputs with data inputs coming in from the client application.
- In addition, outputs from the wrapped program are captured, reformatted, and sent to the client.

Module Level

- Legacy modules are executed using their standard interfaces.
- A significant deviation from the usual calls is that parameters are passed by value – and not by references.
- Therefore, first, the wrapper buffers the received values in its own address space.
- Next, the buffered values are passed on to the invoked module.
- Finally, the output values received from the invoked module are passed on to the client.

Procedural Level

- A procedure internal to the system is invoked as if the procedure was compiled separately.
- As a result, this special treatment of an internal procedure requires:
  1. constructing a parameter interface; and
  2. if needed, initializing global variables before calling the procedure.

Constructing a Wrapper

CS 4423/5523

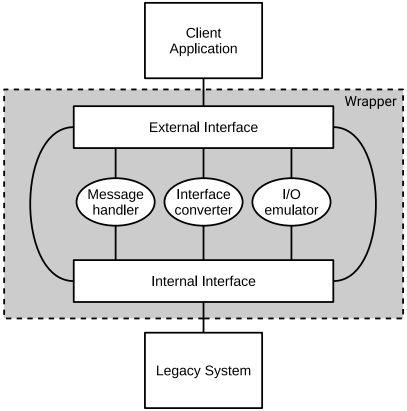
ROAR

Constructing a Wrapper

- A legacy system is wrapped in three steps as follows:
  - A wrapper is constructed.
  - The target program is adapted.
  - The interactions between the target program and the wrapper are verified.
- A wrapper, is a program which receives input messages from the client program, transforms the inputs into an internal representation, and invokes the target system with the newly formatted messages.
- The wrapper intercepts the outputs produced by the target system, transforms them into a format understood by the client, and transfers them to the client.

Constructing a Wrapper

- Conceptually, a wrapper comprises two interfaces and three event driven modules as shown.
- The two interfaces are an internal interface and an external interface, and the three modules are message handler, interface converter, and I/O-emulator.



External Interface

- The interface of the wrapper that is accessed by the clients is called the external interface.
- The wrapper and its client generally communicate by passing messages.
- Messages normally comprise a header and a body.
- The header of a message contains control information:
  1. identity of the sender.
  2. a time stamp.
  3. the transaction code.
  4. target program type.
  5. the identity of the transaction, program, procedure, or module to be invoked.
- The message body contains the input arguments. Similarly, the message body contains the output values in an output message.
- The values contained in a message are normally ASCII strings separated by a delimiter, say, a back slash (“\”).

Internal Interface

- A wrapper’s internal interface is the interface visible to the server.
- The internal interface is dependent upon the language and type of the wrapped entity.
- For example, if the wrapped entity is a job, the internal interface of the wrapper is a job control procedure, and the job control procedure is interpreted to execute the job.
- On the other hand, if the encapsulated entity is a program, procedure, transaction, or a module, the internal interface is a list of parameters in the language of the encapsulated software.
- Therefore, the internal interface is extracted from the source code of the encapsulated software.
- The parameters of the internal interface are specified in terms of the target language, thereby necessitating a translation from the input ASCII strings to the target types.

Message Handler

- The message handler buffers input and output messages, because requests from the client may occasionally arrive at a faster rate than they can be consumed.
- Similarly, outputs from the wrapped program may be produced at a faster rate than they can be sent to the client.

Interface Converter

- This entity converts the internal interface into the external interface and vice-versa.

I/O-emulator

- I/O-emulator intercepts the inputs to and outputs from the wrapped entity.
- Upon intercepting an input, the emulator fills the input buffer with the values of the parameters of the external interface.
- On the other hand, when it intercepts an output value, the emulator copies the contents of the output buffer into the parameter space of the external interface.

Adapting Programs for Wrappers

- A wrapped program is modified to some extent, and it is expected that the modified programs continue to operate in the normal mode.
- Programs are adapted with tools, rather than manually.
- Sneed recommended four types of tools:
  - Transaction wrapper.
  - Program wrapper.
  - Module wrapper.
  - Procedure wrapper.

Idaho State University
Computer Science

Screen Scraping

- Screen scraping is a common form of wrapping in which modern, say, graphical, interfaces replace text-based interfaces.
- The new interface can be a GUI (graphical user interface) or even a web-based HTML (hypertext markup language) client.
- Tools, such as Verastream from Attachmate, automatically generate the new screens.
- Screen scraping simply provides a straightforward way to continue to use a legacy system via a graphical user interface.
- Screen scraping is a short term solution to a larger problem.
- Many serious issues are not addressed by simply mounting a GUI on a legacy system. For example, screen scraping:
  1. does not evolve to support new functions.
  2. incurs high maintenance cost.
  3. ignores the problem of overloading.

For Next Time

- Review EVO Chapter 5.1 - 5.2
- Read EVO Chapter 5.3 - 5.4
- Watch Lecture 11

Are there any questions?