Faculty of Computers and Information
BSc. In Software Engineering Program

Dr. Lamia Abo Zaid

د. لمياء أبوزيد

Software Evolution : TOC

1. Introduction to Software Evolution
2. Taxonomy of Software Maintenance and Evolution
3. Evolution and Maintenance Models
4. Reuse and Domain Engineering
5. Program Comprehension
6. Impact Analysis
7. Refactoring
8. Reengineering
9. Legacy Information Systems

Legacy Software Systems

- Legacy software systems are defined by Bennett as “large software systems that we don’t know how to cope with but that are vital to our organization.”

- Brodie and Stonebraker define a legacy system as “any information system that significantly resists modification and evolution to meet new and constantly changing business requirements”

Legacy Software Systems

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 processor.

Solutions For Legacy

- Legacy systems are mission-critical but 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

1. Freeze.
2. Outsource.
3. Carry on maintenance.
4. Discard and redevelop.
5. **Wrap.**
6. **Migrate.**

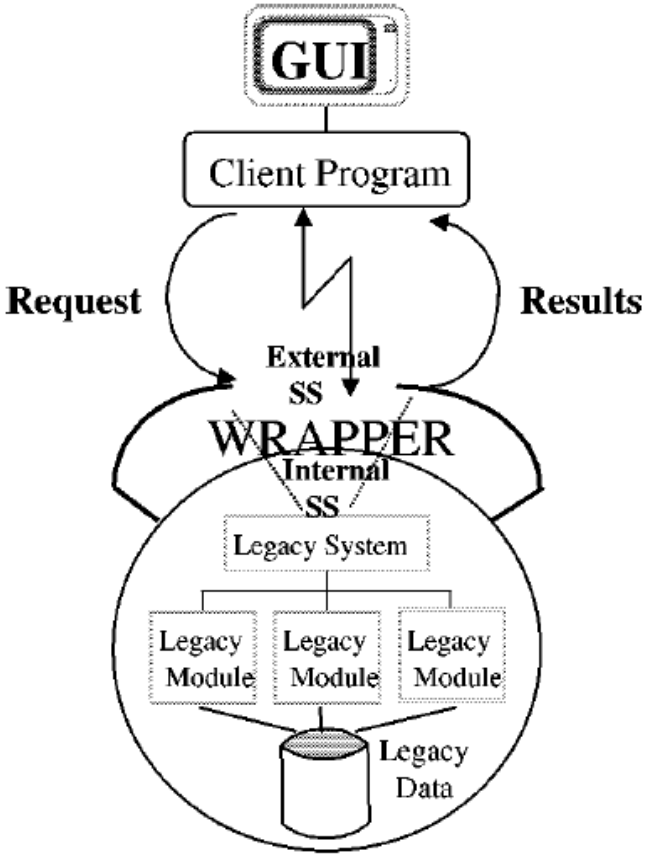
Wrapping

- 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).
- Wrapping is a “black-box” reengineering activity because the interface of the legacy system is analysed but not its internal details

Types of Wrapping

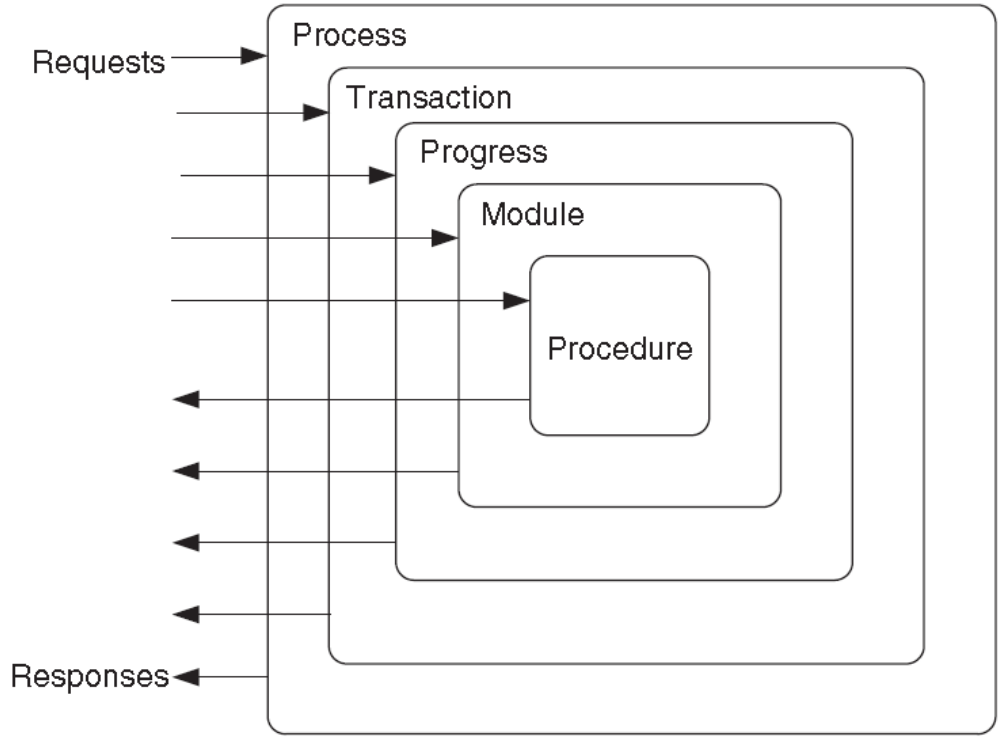
- Orfali et al. [1996] classified wrappers into four categories depending upon what is being wrapped.

1. Database wrappers.
2. System service wrappers.
3. Application wrappers.
4. Function wrappers.



Levels of Encapsulation

- Legacy software has many levels of granularity: procedures are at the lowest level and processes are at the highest level



Constructing a Wrapper

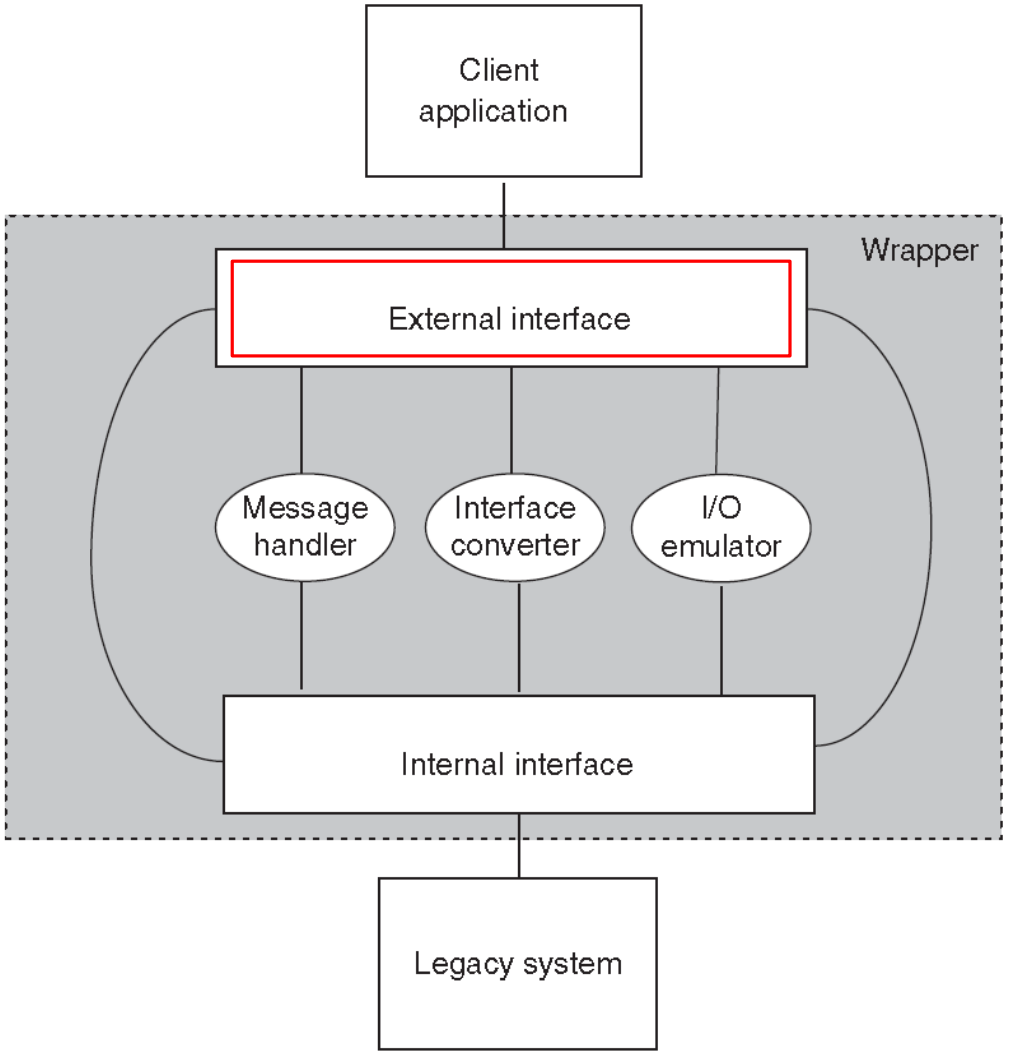
A legacy system is wrapped in three steps :

1. A wrapper is constructed.
2. The target program is adapted.
3. The interactions between the target program and the wrapper are verified.

Constructing a Wrapper

External interface

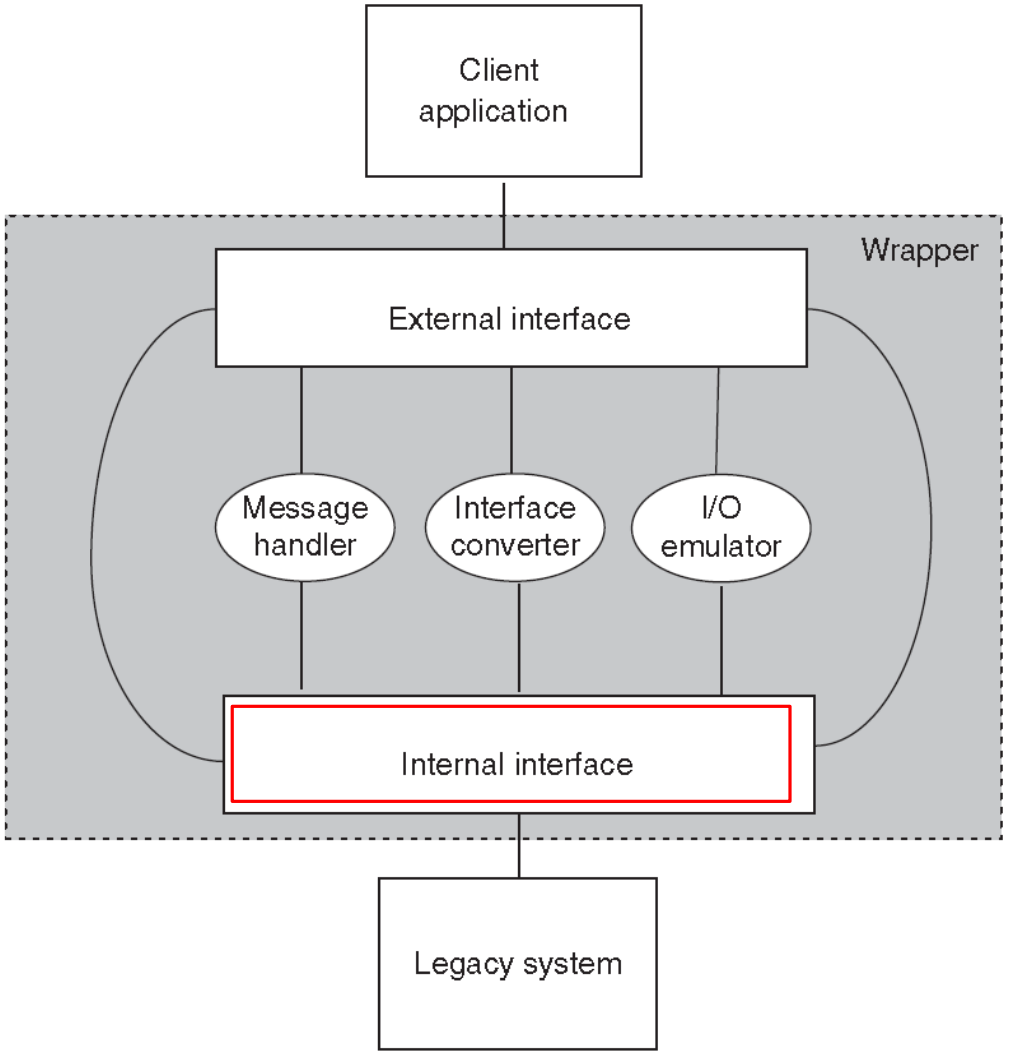
- External interface is the interface of the wrapper that is accessed by the clients. The wrapper and its client generally communicate by passing messages.
- Messages normally comprise a header and a body. The header of a message contains control information:
  - identity of the sender; a time stamp; the transaction code; target program type; 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.



Constructing a Wrapper

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.
- 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.
- 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.



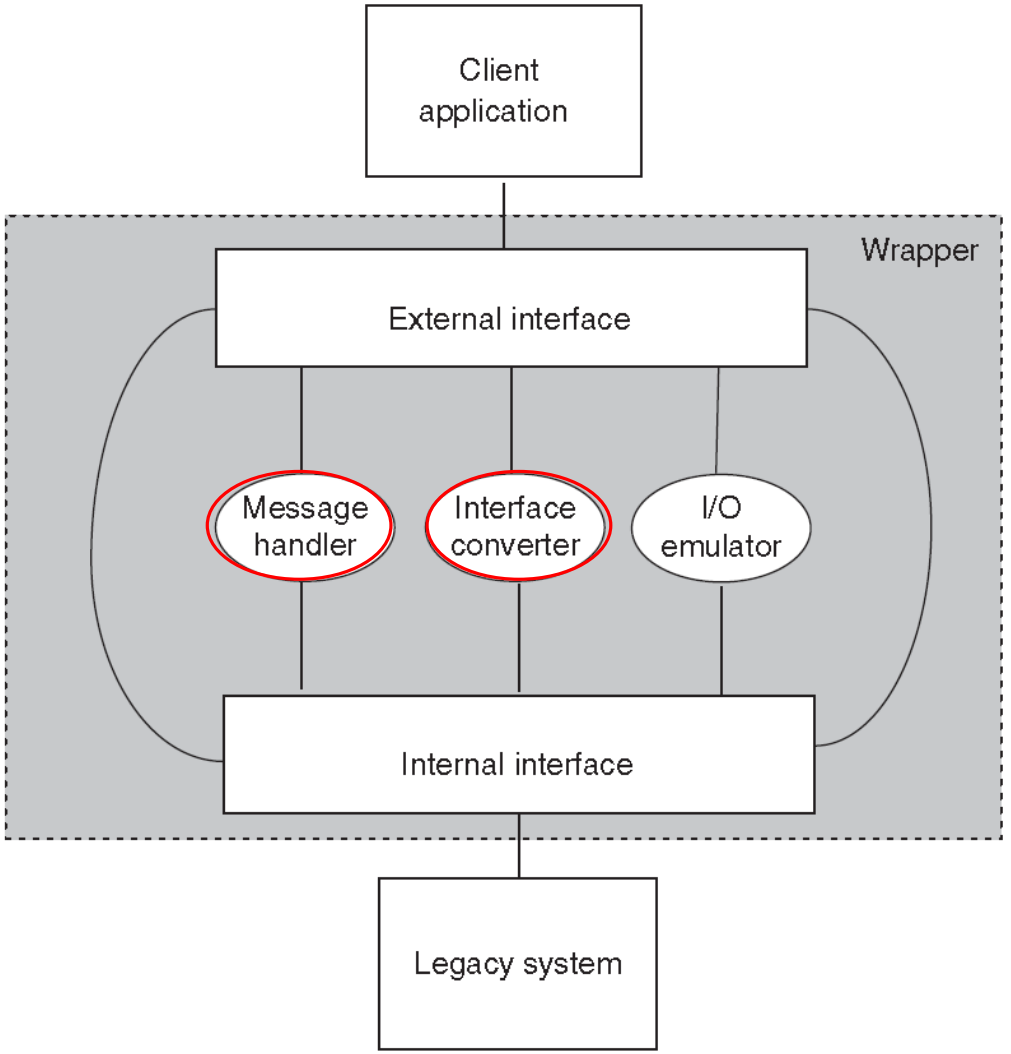
Constructing a Wrapper

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.
- Outputs from the wrapped program may be produced at a faster rate than they can be sent to the client.

Interface converter:

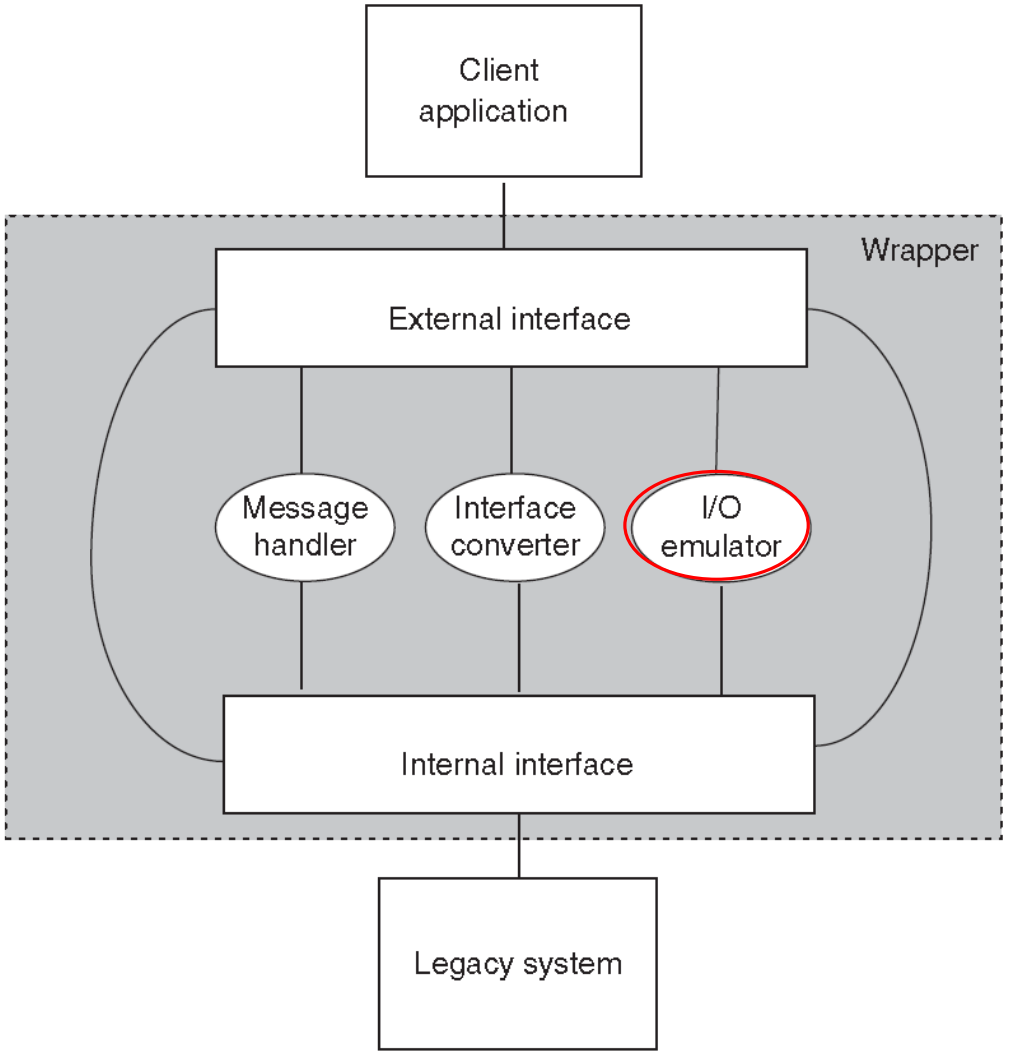
This entity converts the internal interface into the external interface and vice-versa.



Constructing a Wrapper

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.



Wrapping – Screen Scarping

- Screen scraping is a common form of wrapping in which modern interfaces (e.g. graphical, web based) replace text-based interfaces.

- Tools, such as Verastream (https://www.microfocus.com/products/verastream/) automatically generate the new screens

- No new functionality is provided !

Migration

- Migration of legacy systems is the best alternative, when wrapping is unsuitable and redevelopment is not acceptable due to substantial risk.
- Migration involves changes such as: restructuring the system, enhancing the functionality, or modifying the attributes.
- Successful migration gives long-term benefits:
  - Better system understanding
  - Easier maintenance
  - Reduced cost
  - More flexibility to meet future business requirements

Migration

- In general, migration comprises three main steps:

1. Schema Conversion
2. Data Conversion
3. Program Conversion

Migration

1. Schema Conversion

- Schema conversion means translating the legacy database schema into an equivalent database structure expressed in the new technology.
- The transformation of source schema to a target schema is made up of two processes: 1) DBRE which aims to recover the conceptual schema , 2) deriving the target physical schema from this conceptual schema.

Migration

2. Data Conversion

- Data conversion means movement of the data instances from the legacy database to the target database.
- Data conversion requires three steps: extract, transform, and load (ETL).
  1. Extract data from the legacy store.
  2. Transform the extracted data so that their structures match the format. In addition, perform data cleaning (a.k.a. scrubbing or cleansing) to fix or discard data that do not fit the target database.
  3. Load the transformed data in the target database.

Migration

3. Program Conversion

- In the context of LIS migration, program conversion means modifying a program to access the migrated database instead of the legacy data.
- The conversion process leaves the functionalities of the program unchanged.
- Program conversion depends upon the rules that are used in transforming the legacy schema into the target schema.

Migration- Testing and Functionality

- Migration engineers spent close to 80% of their time on testing the target system
- It is important to ensure that the outputs of the target system are consistent with those of the Legacy system
- If both the Legacy system and the target system have the same functionality, it is easier to verify their outputs for compliance.
- However, to justify the project expense and the risk, in practice, migration projects often add new functionalities.

Migration- Testing and Functionality

❑Cut over or Roll over:

- The event of cutting over to the new system from the old one is required to cause minimal disruption to the business process.

❑There are three kinds of transition strategies.

1. Cut-and-Run: The simplest transition strategy is to switch off the legacy system and turn on the new system.
2. Phased Interoperability: To reduce risks, cut over is gradually performed in incremental steps.
3. Parallel Operation: The target system and the LIS operate at the same time. Once the new system is considered to be reliable, the LIS is taken off service.

Migration Planning

The activity of planning for migration comprises thirteen steps:

Step 1: Perform portfolio analysis.

Step 2: Identify the stakeholders.

Step 3: Understand the requirements.

Step 4: Create a business case.

Step 5: Make a go or no-go decision.

Step 6: Understand the LIS.

Step 7: Understand the target technology.

Step 8: Evaluate the available technologies.

Step 9: Define the target architecture.

Step 10: Define a strategy.

Step 11: Reconcile the strategy with the needs of the stakeholder.

Step 12: Determine the resources required.

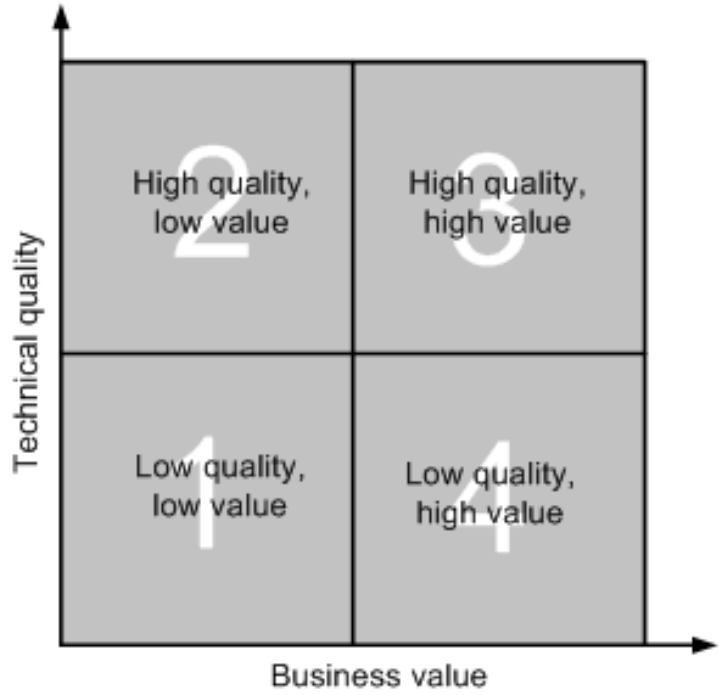
Step 13: Evaluate the feasibility of the strategy.

Migration Planning

Step 1: Perform portfolio analysis.

- Portfolio analysis establishes measures of technical quality and business value for a set of software systems. It is represented on a chi-square chart

1. Quadrant 1: A system with low technical quality and low business value is a prime candidate for substitution with a commercial product, if one is available.
2. Quadrant 2: A system with high technical quality but low business value need not be replaced, migrated, or restructured.
3. Quadrant 3: A system with high technical quality and high business value should be actively evolved.
4. Quadrant 4: A system with low technical quality but high business value is a good candidate for redevelopment or migration to a new system.



Migration Planning

Step 2: Identify the stakeholders.

- Stakeholders are the people or organizations that influence a system's behavior, or those who are impacted by the system.
- Stakeholders typically include architects, developers, maintainers, managers, customers, and end users.
- The stakeholders ultimately judge the outcome and impact of the migration project.
- Each of the stakeholders bring their own perspectives to the table.
- Therefore, it is necessary to obtain their agreement and support.

Migration Planning

Step 3: Understand the requirements.

- There are two challenges in defining requirements.
  - First, ensure that the right requirements are captured.
  - Second, express the requirements in such a way that the stakeholders can easily review and confirm their correctness.
- Requirements can come from:
  - Legacy Information System
  - Business Process Reengineering
  - Stakeholders

Migration Planning

Step 4: Create a business case.

- Based on a business case, executive management can decide whether or not the migration project will increase quality, reduce maintenance costs, and be financially viable.
- A good business case provides the following information about the migration project:
  - Problem statement; Solution; Risks; and Benefits.

Objective	Sample quantifiable benefit metrics
Lower maintenance cost	Average cycle time to close problem reports
	Average labor hours to close problem reports
	Total staff census
	Average problem-report backlog
	Post-release fix rework hours
Add new functionality	Count of new functions added to the product
	Value added or revenue generated by new functions
Increase performance	Number of delivered operations, such as transactions, per unit time
Replace old equipment	Net annualized cost of purchase and maintenance
Recode in different languages	Number of modules in each programming language
Reuse of existing artifacts	Number of artifacts used in other products
Data rationalization	Number of redundant database objects removed
Integrate disjoint applications	Number of unified applications accessible to users
	Measures of usability and training time required for application suite

Migration Planning

Step 5: Make a go or no-go decision.

- Once a business case has been defined, it is reviewed by the stakeholders to reach an agreement.
- If the business case is unsatisfactory then the legacy migration project is terminated at this step.

Step 6: Understand the legacy system.

- Understanding the LIS is essential to the success of any migration project.
- Techniques available to meet this challenge include program comprehension and reverse engineering.

Migration Planning

Step 7: Understand the target technology.

- It is important to understand the technologies that can be used in the migration effort and the technologies that have been used in the legacy system.
- In general, four types of technologies are of interest in the migration effort:

1. Languages and DBMS available, including COBOL, Java, eXtensible Markup Language (XML), and modern DBMS.
2. Distributed transaction models, including distributed communication and transaction technologies such as remote procedure calls (RPC) or message queues.
3. Middleware technologies and standards that may be used to develop an information system, including message-oriented middleware (MOM), XML Messaging, Java 2 Enterprise Edition (J2EE), and Enterprise JavaBeans (EJB).
4. Tools that are available to assist in migration of the LIS to the new information system.

Migration Planning

Step 8: Evaluate the available technologies.

- One must compare and contrast all the available technologies to evaluate their capabilities.
- To formulate the eventual architecture and design of the system, evaluations are performed.

Step 9: Define the target architecture.

- The target architecture is the desired architecture of the new system.
- It models the stakeholders' vision of the new system. This usually requires descriptions using different views with different levels of granularity.
- The target architecture is likely to evolve during the migration process. It is continually reevaluated and updated during the migration process.

Migration Planning

Step 10: Define a strategy.

- A strategy defines the overall process of transforming the LIS to the new system.
- This includes migration methodology, that is schema conversion, data conversion, and program conversion, testing, and cut over.
- For a mission-critical legacy system, deploying the new system all at once is a risky procedure, therefore a legacy system is evolved incrementally to the new system.
- During the migration effort, many things can change: user requirements may change, additional knowledge about the system may be acquired, and the technology may change.
- Those changes must be accommodated by the migration effort.
- While accommodating those changes, a migration strategy need to minimize risk, minimize development and deployment costs, support an aggressive but reliable schedule, and meet system quality expectations.

Migration Planning

Step 11: Reconcile the strategy with the needs of the stakeholder.

- A The migration strategy developed in the previous step must be reconciled with stakeholder needs.
- Therefore, this step includes briefing the stakeholders about the approach, reviewing the target architecture, and the strategy.
- The entire group evaluates the strategy and provides input for the final consensus profile.

Step 12: Determine the resources required.

- We estimate the resource need including cost of implementing the project.
- One can use the widely used cost estimation model called Constructive Cost Model II (COCOMO II).
- COCOMO II addresses nonsequential process models, reengineering work, and reuse-driven approach.
- The COCOMO II model provides estimates of effort, schedule by phases, and staffing by phases and activities.

Migration Planning

Step 13: Evaluate the feasibility of the strategy.

- After executing the first 12 steps, the management should have an understanding of the system under migration, the available technology options, a target architecture, migration strategy, cost of migration, and a schedule to effect migration.
- Based on available information, management determines whether or not the migration strategy is feasible.
- If the strategy is found to be viable, the migration plan is finalized.
- On the other hand, if it is unacceptable, a detailed report is produced.
- Based on the reasons stated in the report, one may revise the migration strategy until:
  - A feasible approach can be identified, or
  - The migration strategy is determined to be infeasible and the project is terminated.

Questions

?