System Reengineering

Objectives

- To explain why software re-engineering is a cost-effective option for system evolution
- To describe the activities involved in the software re-engineering process
- To distinguish between software and data re-engineering and to explain the problems of data re-engineering

Software re-engineering

- Reorganising and modifying existing software systems to make them more maintainable

- "the examination of a subject system to reconstitute it in a new form and the subsequent implementation of the new form.

  [ElliotChikofsky and JamesCross, Reverse Engineering and Design Recovery: A Taxonomy, IEEE Software 7(1):13-17, 1990.]

Topics covered

- Source code translation
- Reverse engineering
- Program structure improvement
- Program modularisation
- Data re-engineering

System re-engineering

- Re-structuring or re-writing 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
- Re-engineering involves adding effort to make
  them easier to maintain. The system
  may be re-structured and re-

When to re-engineer

- When system changes are mostly confined to
part of the system then re-engineer that part
- When hardware or software support becomes
obsolete
- When new ways of accessing are needed, but its functionality remains
- When tool support is are available

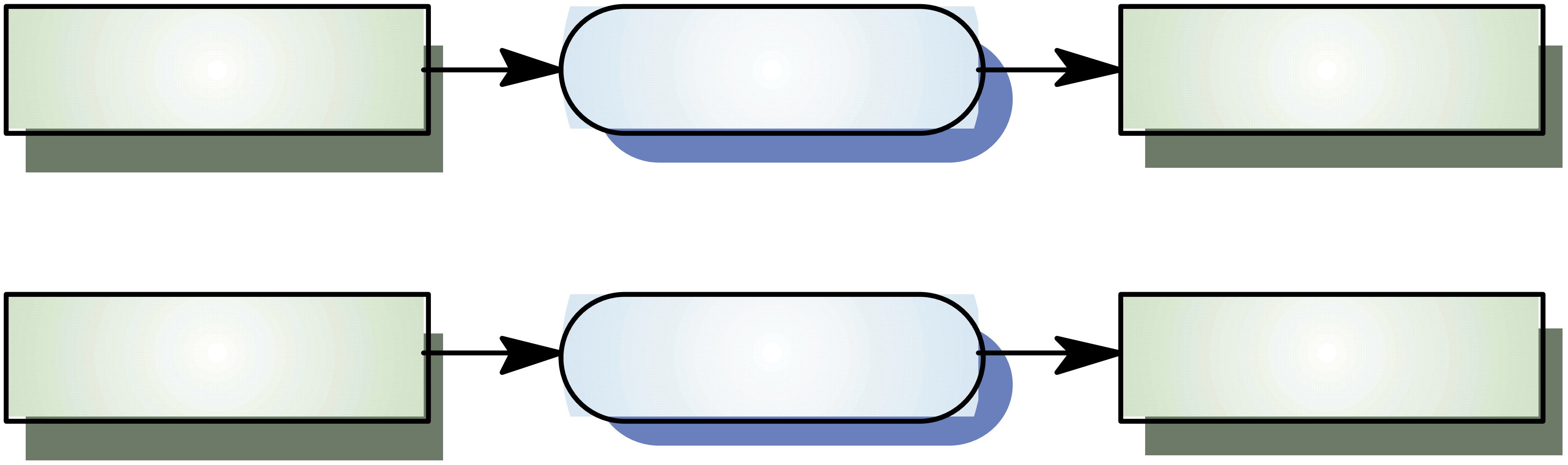
Re-engineering advantages

- 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

Business process re-engineering

- Concerned with re-designing business processes to make them more responsive and more efficient
- Often reliant on the introduction of new computer systems to support the revised processes
- May force software re-engineering as the legacy systems are designed to support existing processes

Forward engineering and re-engineering

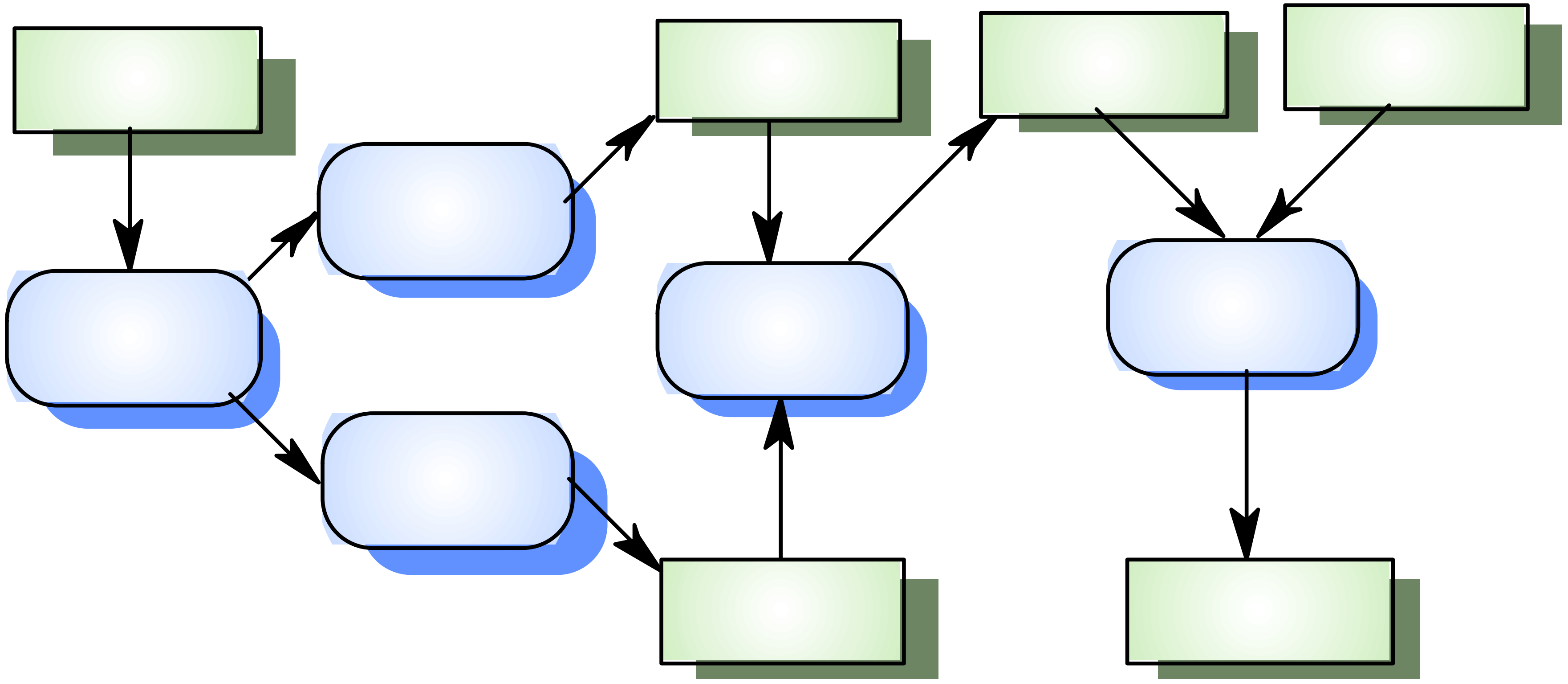


Forward engineering and re-engineering

“**Forward engineering** is the traditional process of moving from high-level abstractions and logical, implementation-independent designs to the physical implementation of a system."

[ElliotChikofsky and JamesCross, Reverse Engineering and Design Recovery: A Taxonomy, IEEE Software 7(1):13-17, 1990.]

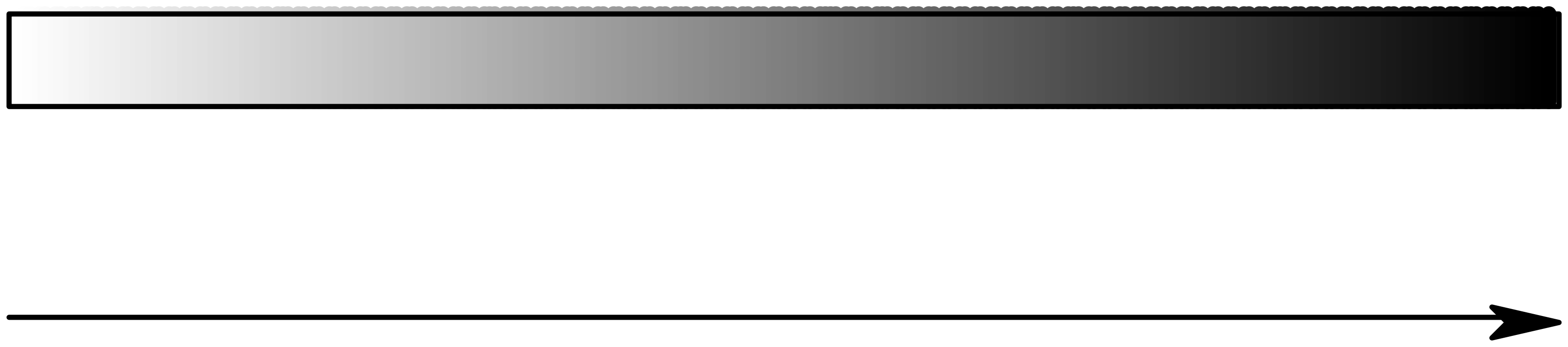
The re-engineering process



Re-engineering cost factors

- The *quality* of the software to be re-engineered
- The *tool support* available for re-engineering
- The *extent of the data conversion* which is required
- The availability of *expert staff* for re-engineering

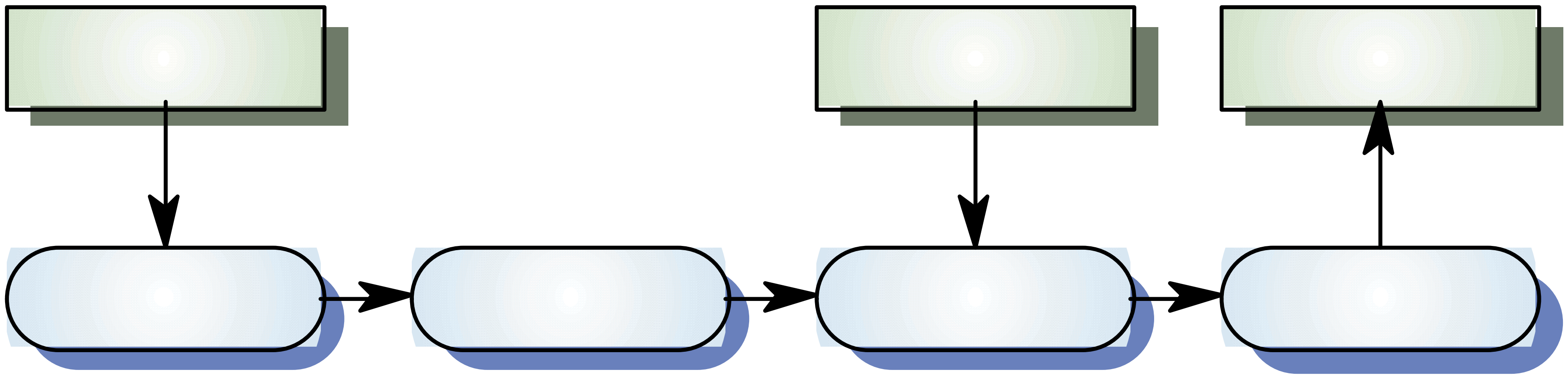
Re-engineering approaches



Source code translation

- Involves converting the code from one language (or language version) to another e.g. FORTRAN to C
- May be necessary because of:
  - Hardware platform update
  - Staff skill shortages
  - Organisational policy changes
- Only realistic if an automatic translator is available

The program translation process



Reverse engineering

- Analysing software with a view to understanding its design and specification
- May be part of a re-engineering process but may also be used to re-specify a system for re-implementation
- Builds a program data base and generates information from this
- Program understanding tools (browsers, cross-reference generators, etc.) may be used in this process

Reverse engineering

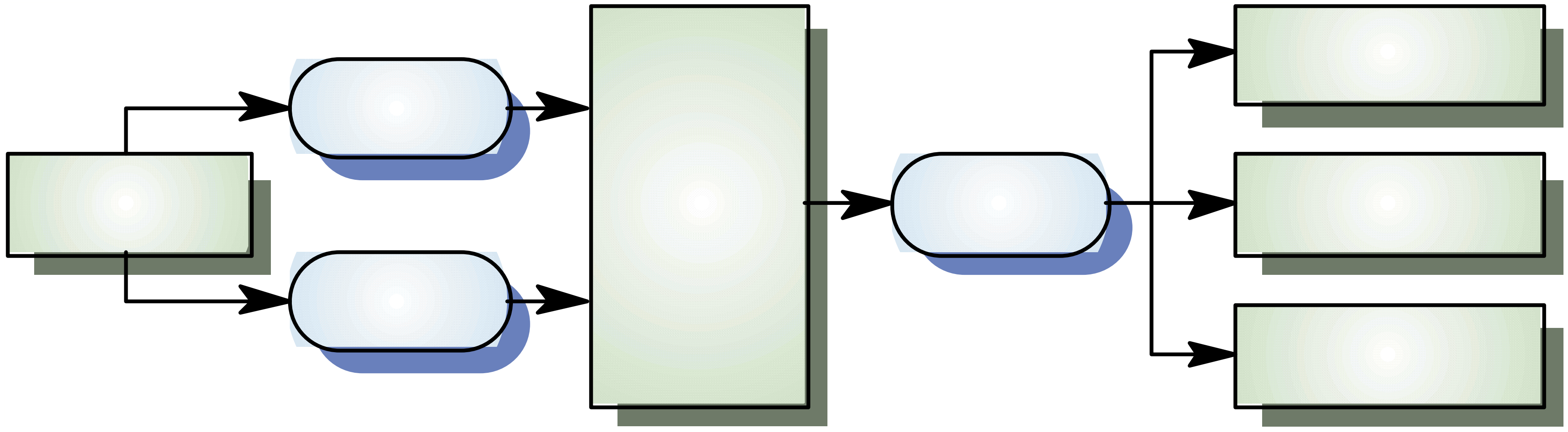
- “**Reverse engineering** is the process of analyzing a subject system with two goals in mind:

(1) to identify the system's components and their interrelationships; and,

(2) to create representations of the system in another form or at a higher level of abstraction."

[ElliotChikofsky and JamesCross, Reverse Engineering and Design Recovery: A Taxonomy, IEEE Software 7(1):13-17, 1990.]

The reverse engineering process



Reverse engineering

- "**Design recovery** is a subset of reverse engineering in which domain knowledge, external information, and deduction or fuzzy reasoning are added to the observations of the subject system."
- The objective of design recovery is to identify meaningful higher-level abstractions beyond those obtained directly by examining the system itself.
- [ElliotChikofsky and JamesCross, Reverse Engineering and Design Recovery: A Taxonomy, IEEE Software

Reverse engineering

- Reverse engineering often precedes re-engineering but is sometimes worthwhile in its own right
  - The design and specification of a system may be reverse engineered so that they can be an input to the requirements specification process for the system’s replacement
  - The design and specification may be reverse engineered to support program maintenance and

Program structure improvement

- Maintenance tends to corrupt the structure of a program. It becomes harder and harder to understand
- The program may be automatically restructured to remove unconditional branches
- Conditions may be simplified to make them more readable

Program Restructuring

- "**Restructuring** is a transformation from one form of representation to another at the same relative level of abstraction." The new representation is meant to preserve the semantics and external behaviour of the original.

  [ElliotChikofsky and JamesCross, Reverse Engineering and Design Recovery: A Taxonomy, IEEE Software 7(1):13-17, 1990.]

Spaghetti logic

Structured control logic

Another Spaghetti logic

```
START:
              GOTO MAMMALS
DOG:
               WALK THE DOG
               GOTO CAT
MAMMALS:
               GOTO DOG
FISH:
               FEED THE FISH
               COVER THE BIRD
               GOTO FROG
CAT:
                PUT OUT THE CAT
                GOTO FISH-AND-FOWL
FISH-AND-FOWL:
                GOTO FISH
FROG:
               SING TO THE FROG
EXIT.
```

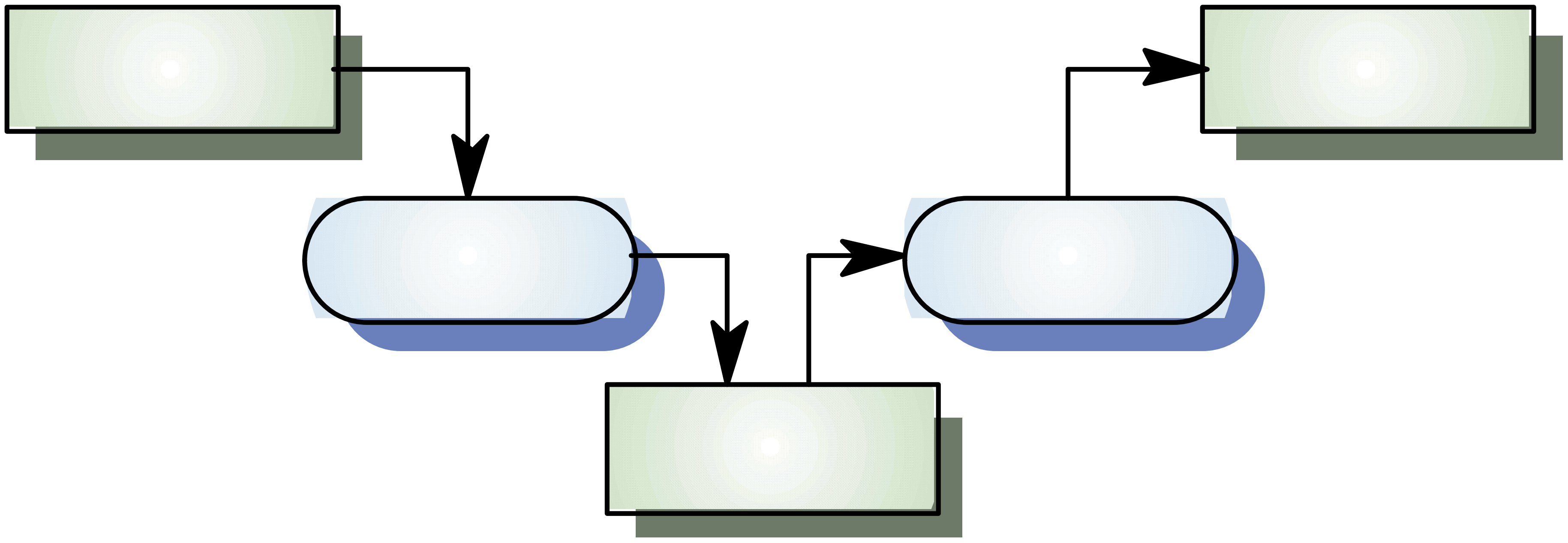
Another structured control logic

```
START:
   CALL FUNCTION  DOG
   CALL FUNCTION  CAT
   CALL FUNCTION  FISH
   CALL FUNCTION  BIRD
   CALL FUNCTION  FROG
  EXIT.

DOG:
   WALK THE DOG
   RETURN

CAT:
   PUT OUT THE CAT
   RETURN

FISH:
   FEED THE FISH
   RETURN

BIRD:
   COVER BIRD CAGE
   RETURN

FROG:
   SING TO THE FROG
   RETURN
```

Condition simplification

```
-- Complex condition
if not (A > B and (C < D or not ( E > F) ) )...

-- Simplified condition
if (A <= B and (C>= D or E > F)...
```

Automatic program restructuring



Restructuring problems

- Problems with re-structuring are:
  - Loss of comments
  - Loss of documentation
  - Heavy computational demands
- Restructuring doesn’t help with poor modularisation where related components are dispersed throughout the code
- The understandability of data-driven programs may not be improved by re-structuring

Program modularisation

- The process of re-organising a program so that related program parts are collected together in a single module
- Usually a manual process that is carried out by program inspection and re-organisation

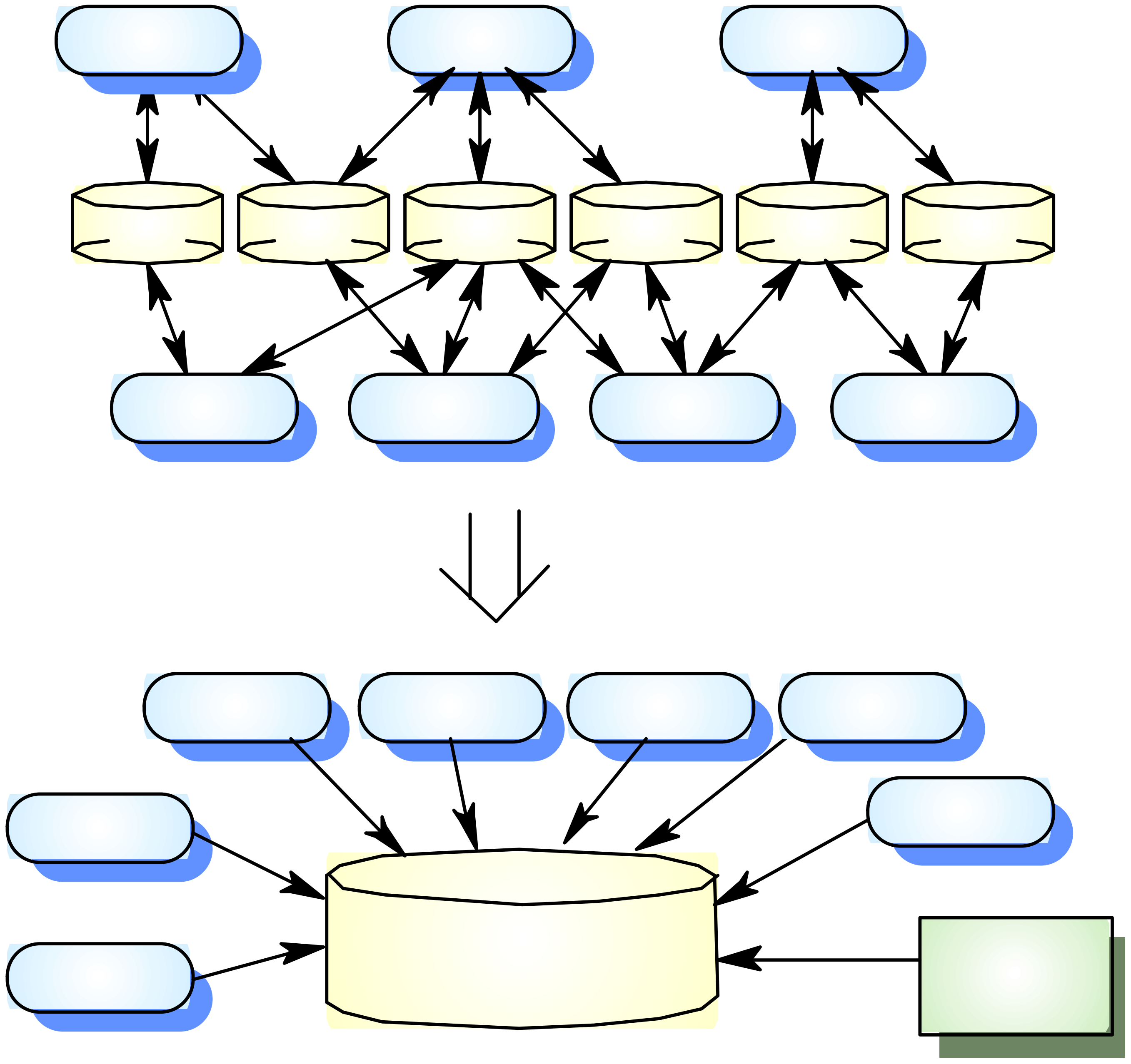
Data re-engineering

- Involves analysing and reorganising the data structures (and sometimes the data values) in a program
- May be part of the process of migrating from a file-based system to a DBMS-based system or changing from one DBMS to another
- Objective is to create a managed data environment

Approaches to data re-engineering

Data problems

- End-users want data on their desktop machines rather than in a file system. They need to be able to download this data from a DBMS
- Systems may have to process much more data than was originally intended by their designers
- Redundant data may be stored in different formats in different places in the system



Data migratio
n

Data problems

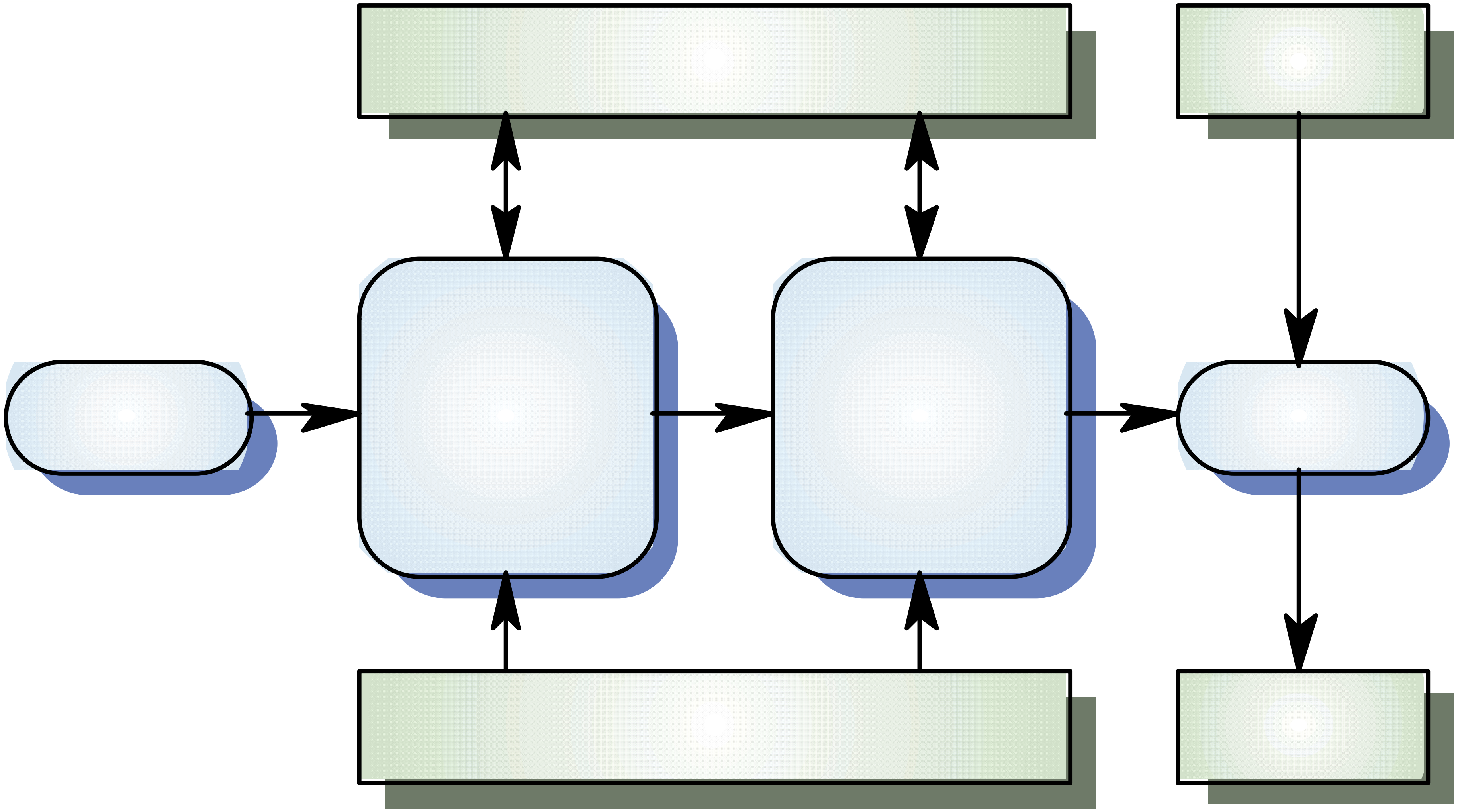
- Data naming problems
  - Names may be hard to understand. The same data may have different names in different programs
- Field length problems
  - The same item may be assigned different lengths in different programs
- Record organisation problems
  - Records representing the same entity may be organised differently in different programs
- Hard-coded literals
- No data dictionary

Data value inconsistencies

Data conversion

- Data re-engineering may involve changing the data structure organisation without changing the data values
- Data value conversion is very expensive. Special-purpose programs have to be written to carry out the conversion

The data re-engineering process



Key points

- The objective of re-engineering is to improve the system structure to make it easier to understand and maintain
- The re-engineering process involves source code translation, reverse engineering, program structure improvement, program modularisation and data re-engineering
- Source code translation is the automatic conversion of of program in one language to another

Key points

- Reverse engineering is the process of deriving the system design and specification from its source code
- Program structure improvement replaces unstructured control constructs with while loops and simple conditionals
- Program modularisation involves reorganisation to group related items
- Data re-engineering may be necessary because of inconsistent data management