Chapter 11
Application Architectures

Objectives

- To explain the organisation of two fundamental models of business systems - batch processing and transaction processing systems
- To describe the abstract architecture of resource management systems
- To explain how generic editors are event processing systems
- To describe the structure of language processing systems

Topics covered

- 11.1 Data processing systems
- 11.2 Transaction processing systems
- 11.3 Event processing systems
- 11.4 Language processing systems

Generic application architectures

- Application systems are designed to meet an organisational need.
- As businesses have much in common, their application systems also tend to have a common architecture that reflects the application requirements.
- A generic architecture is configured and adapted to create a system that meets specific requirements.

Use of application architectures

- As a starting point for architectural design.
- As a design checklist.
- As a way of organising the work of the development team.
- As a means of assessing components for reuse.
- As a vocabulary for talking about application types.

Application types

- Data processing applications
  - Data driven applications that process data in batches without explicit user intervention during the processing.
- Transaction processing applications
  - Data-centred applications that process user requests and update information in a system database.
- Event processing systems
  - Applications where system actions depend on interpreting events from the system’s environment.
- Language processing systems
  - Applications where the users’ intentions are specified in a formal language that is processed and interpreted by the system.

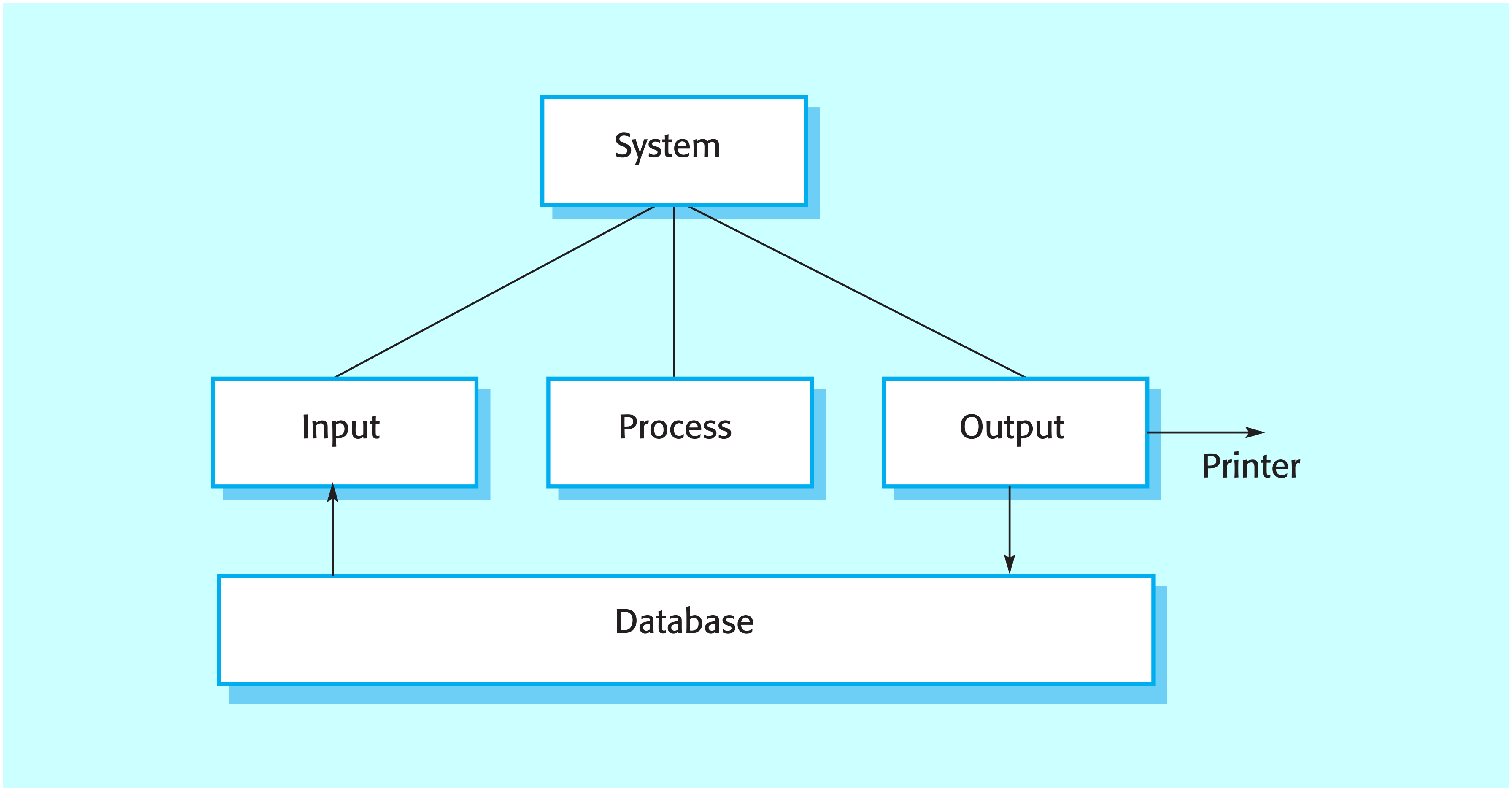
Application type examples

- Data processing systems
  - Billing systems;
  - Payroll systems.
- Transaction processing systems
  - E-commerce systems;
  - Reservation systems.
- Event processing systems
  - Word processors;
  - Real-time systems.
- Language processing systems
  - Compilers;
  - Command interpreters.

Data processing systems

- Systems that are data-centred where the databases used are usually orders of magnitude larger than the software itself.
- Data is input and output in batches
  - Input: A set of customer numbers and associated readings of an electricity meter;
  - Output: A corresponding set of bills, one for each customer number.
- Data processing systems usually have an input-process-output structure.

Input-process-output model



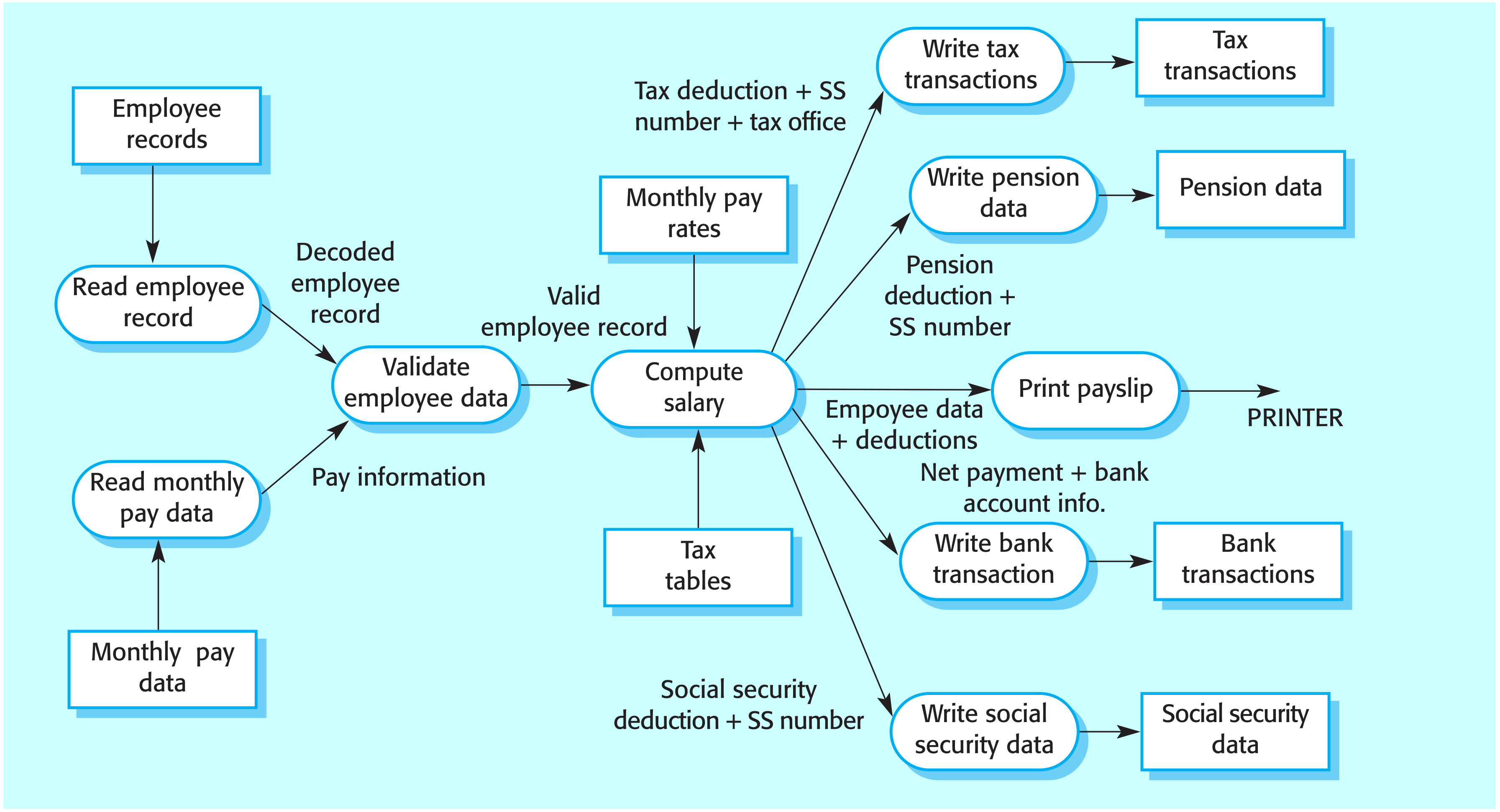
Input-process-output

- The input component reads data from a file or database, checks its validity and queues the valid data for processing.
- The process component takes a transaction from the queue (input), performs computations and creates a new record with the results of the computation.
- The output component reads these records, formats them accordingly and writes them to the database or sends them to a printer.

Data-flow diagrams

- Show how data is processed as it moves through a system.
- Transformations are represented as round-edged rectangles, data-flows as arrows between them and files/data stores as rectangles.

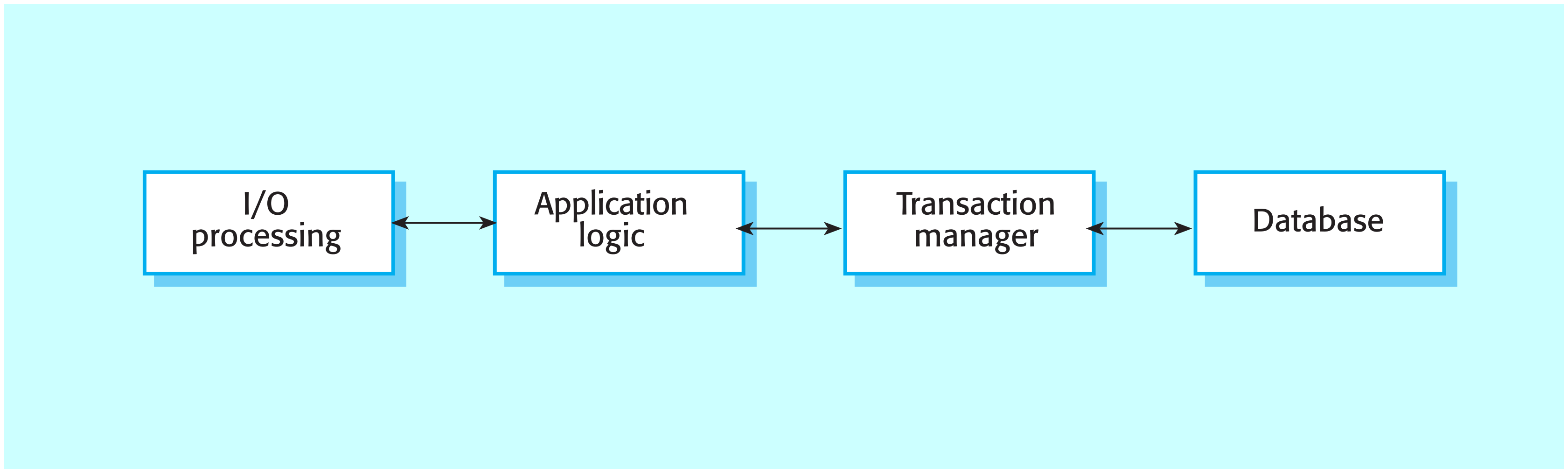
Salary payment DFD



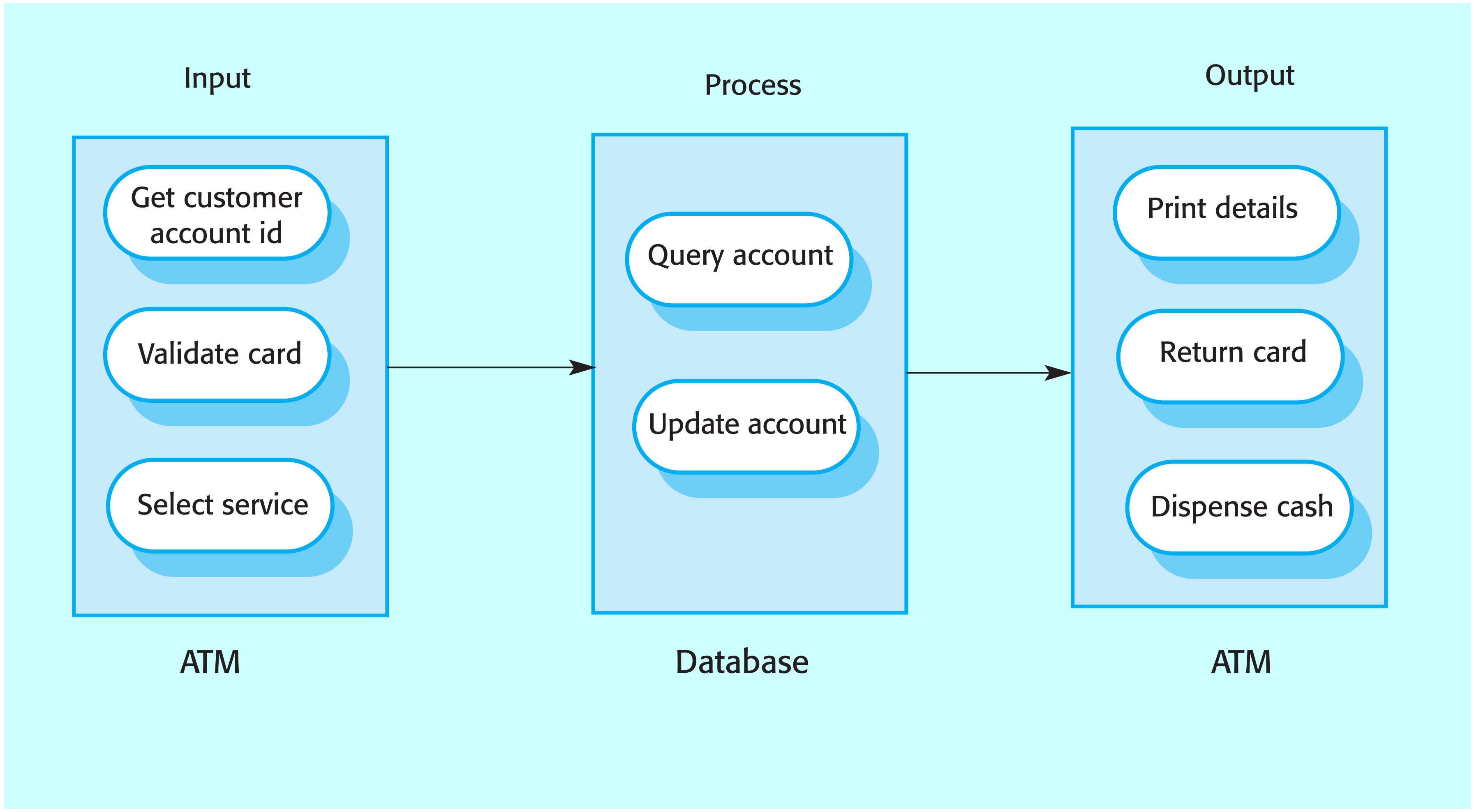
Transaction processing systems

- Process user requests for information from a database or requests to update the database.
- From a user perspective a transaction is:
  - Any coherent sequence of operations that satisfies a goal;
  - For example - find the times of flights from London to Paris.
- Users make asynchronous requests for service which are then processed by a transaction manager.

Transaction processing



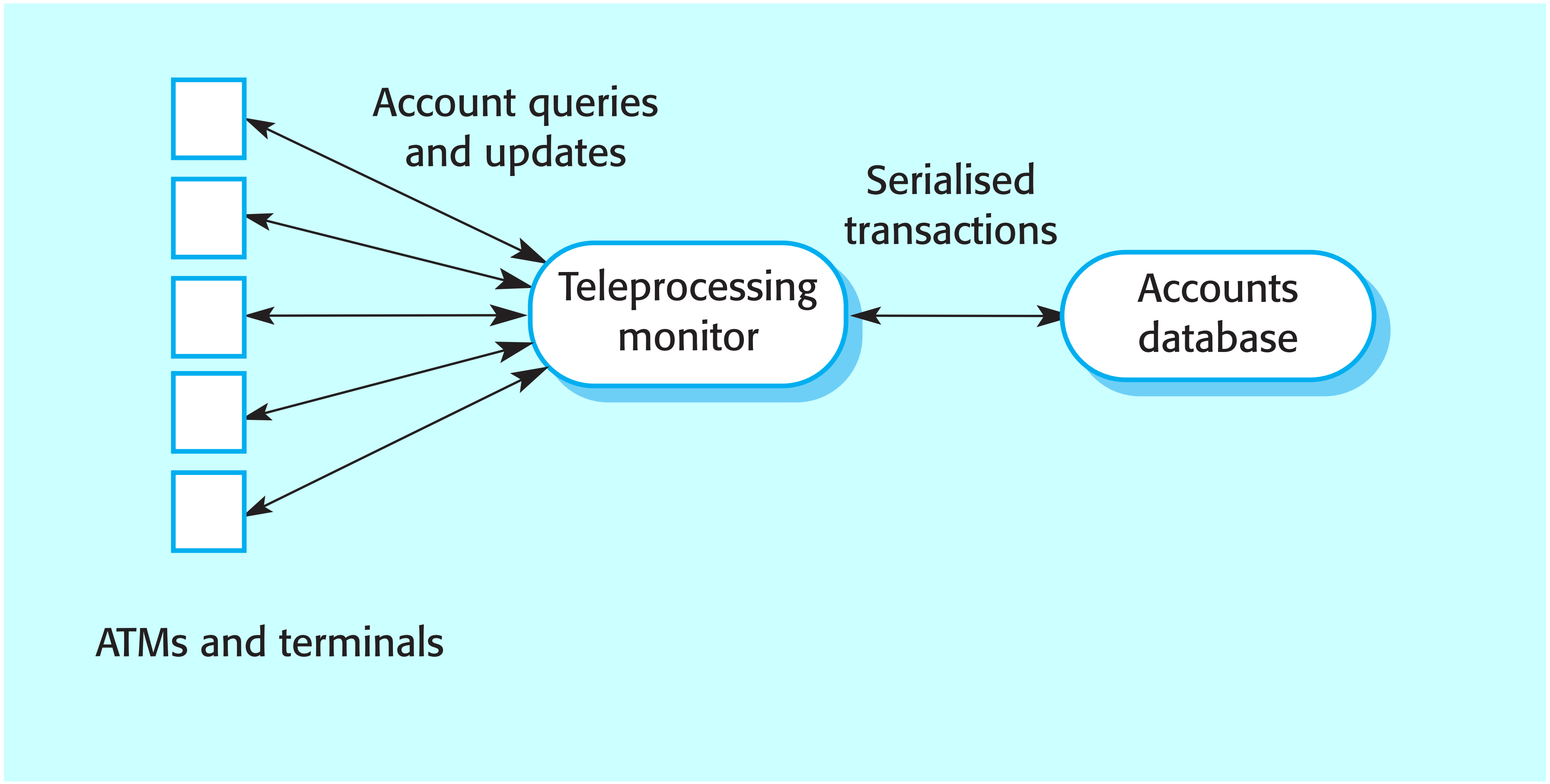
ATM system organisation



Transaction processing middleware

- Transaction management middleware or teleprocessing monitors handle communications with different terminal types (e.g. ATMs and counter terminals), serialises data and sends it for processing.
- Query processing takes place in the system database and results are sent back through the transaction manager to the user’s terminal.

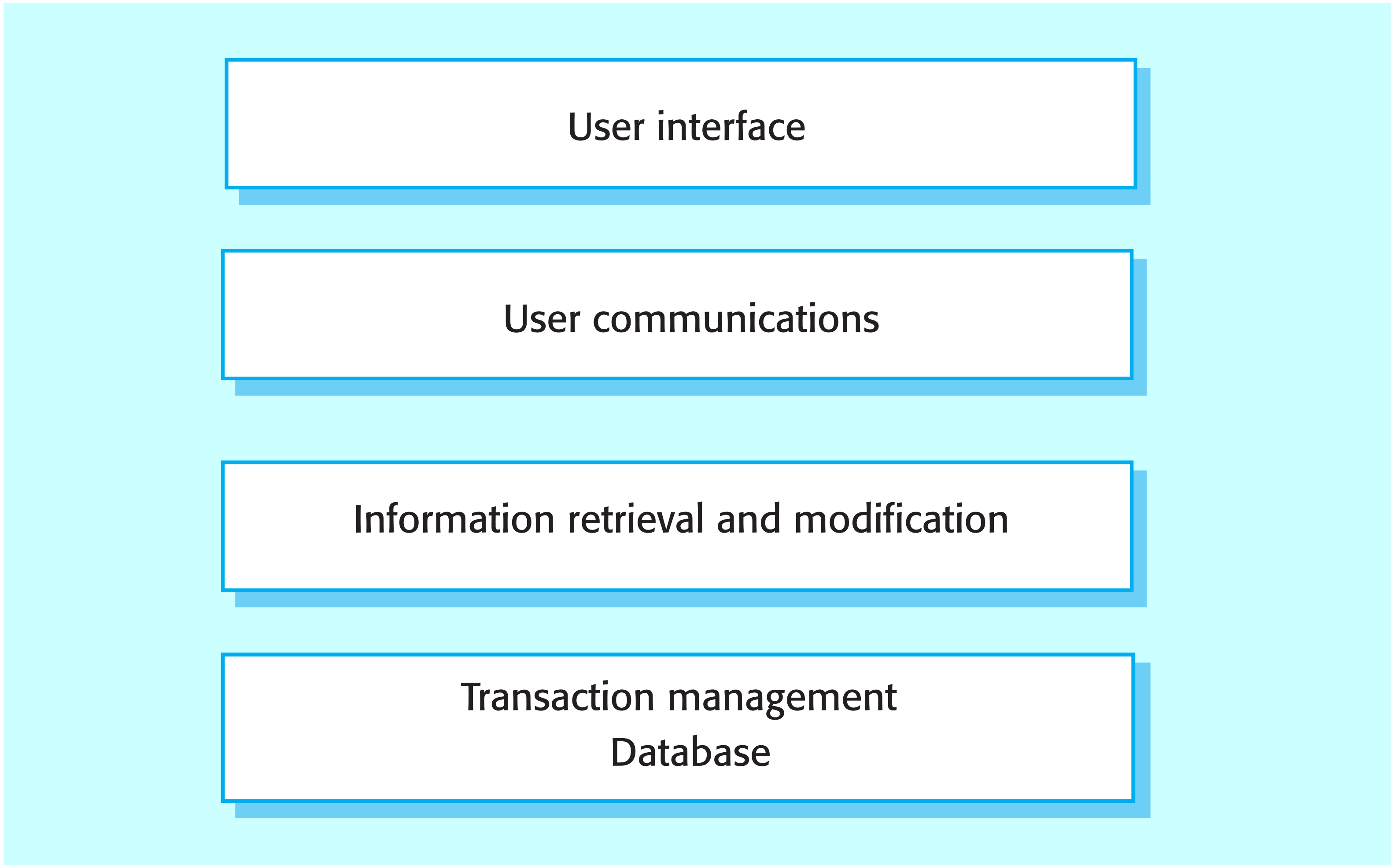
Transaction management



Information systems architecture

- Information systems have a generic architecture that can be organised as a layered architecture.
- Layers include:
  - The user interface
  - User communications
  - Information retrieval
  - System database

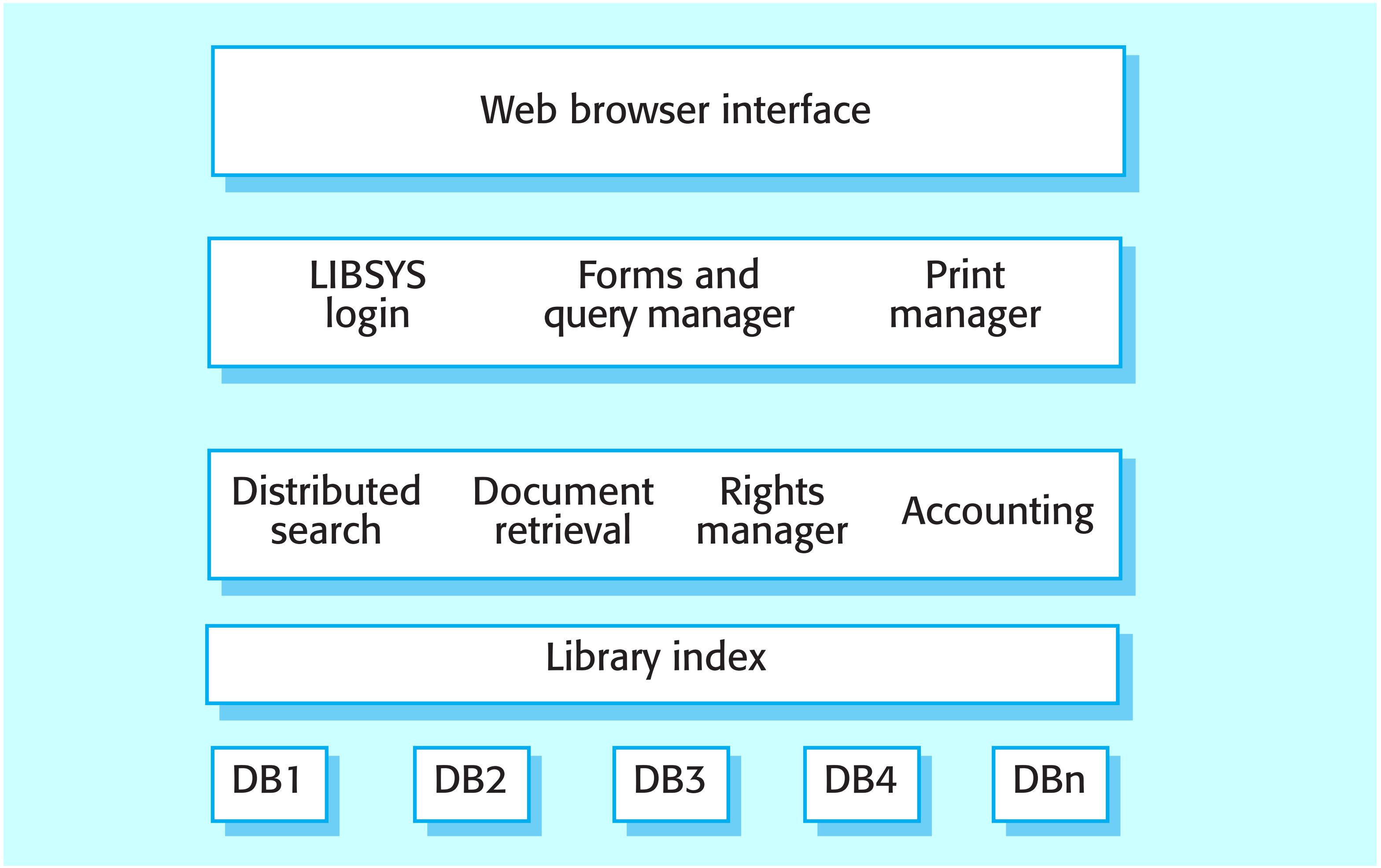
Information system structure



LIBSYS architecture

- The library system LIBSYS is an example of an information system.
- User communications layer:
  - LIBSYS login component;
  - Form and query manager;
  - Print manager;
- Information retrieval layer
  - Distributed search;
  - Document retrieval;
  - Rights manager;
  - Accounting.

LIBSYS organisation



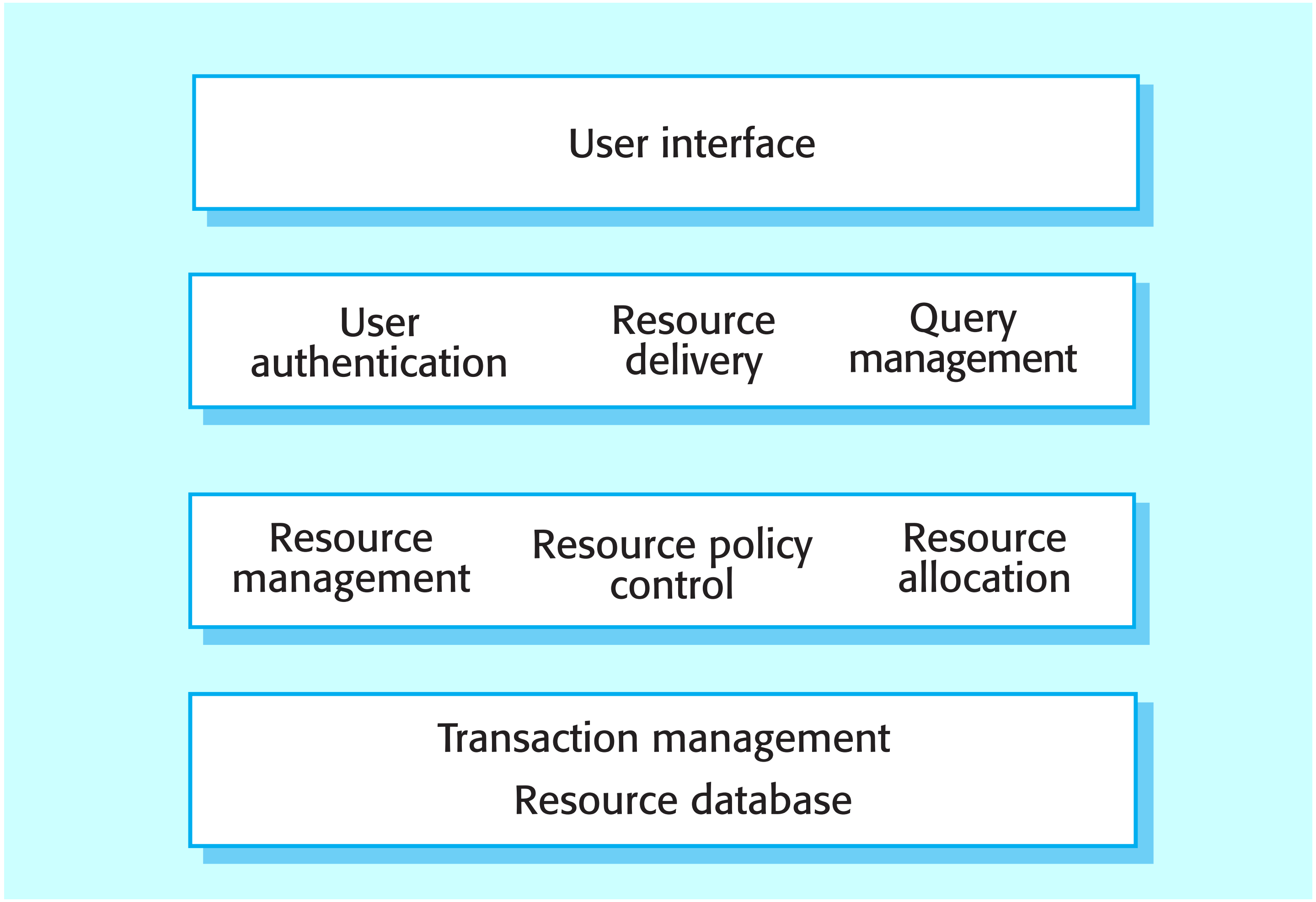
Resource allocation systems

- Systems that manage a fixed amount of some resource (football game tickets, books in a bookshop, etc.) and allocate this to users.
- Examples of resource allocation systems:
  - Timetabling systems where the resource being allocated is a time period;
  - Library systems where the resource being managed is books and other items for loan;
  - Air traffic control systems where the resource being managed is the airspace.

Resource allocation architecture

- Resource allocation systems are also layered systems that include:
  - A resource database;
  - A rule set describing how resources are allocated;
  - A resource manager;
  - A resource allocator;
  - User authentication;
  - Query management;
  - Resource delivery component;
  - User interface.

Layered resource allocation

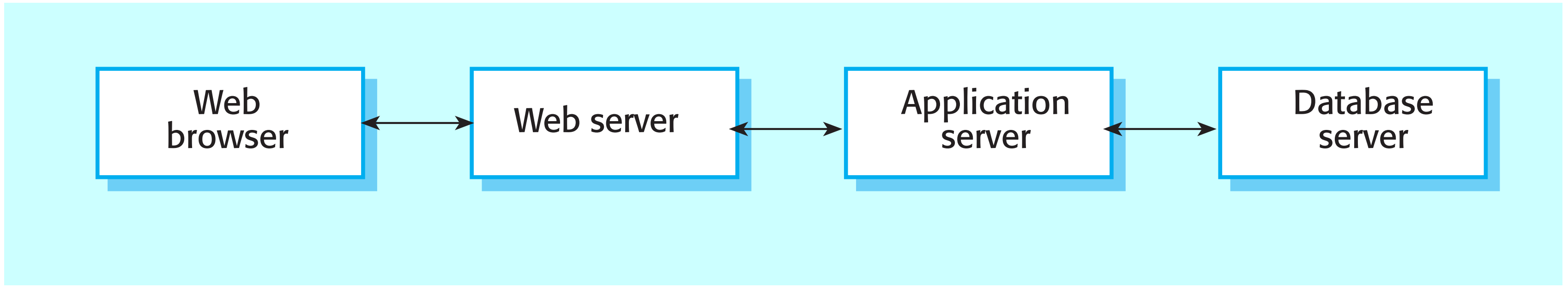


Layered system implementation

- Each layer can be implemented as a large scale component running on a separate server. This is the most commonly used architectural model for web-based systems.
- On a single machine, the middle layers are implemented as a separate program that communicates with the database through its API.
- Fine-grain components within layers can be implemented as web services.

E-commerce system architecture

- E-commerce systems are Internet-based resource management systems that accept electronic orders for goods or services.
- They are usually organised using a multi-tier architecture with application layers associated with each tier.



Event processing systems

- These systems respond to events in the system’s environment.
- Their key characteristic is that event timing is unpredictable so the architecture has to be organised to handle this.
- Many common systems such as word processors, games, etc. are event processing systems.

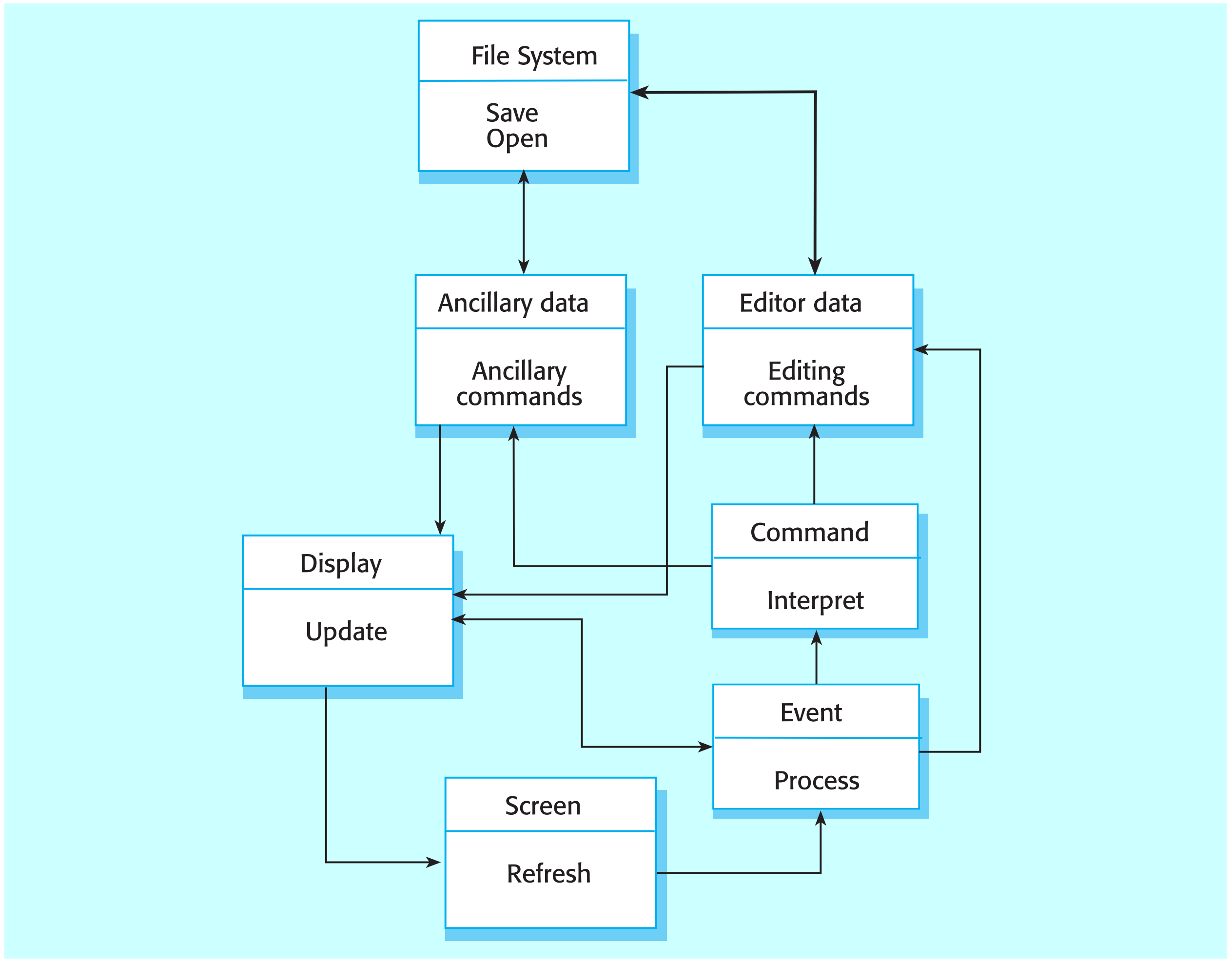
Editing systems

- Real-time systems (Chapter 15) and editing systems are the most common types of event processing system.
- Editing system characteristics:
  - Single user systems;
  - Must provide rapid feedback to user actions;
  - Organised around long transactions so may include recovery facilities.

Editing system components

- Editing systems are naturally object-oriented:
  - Screen - monitors screen memory and detects events;
  - Event - recognises events and passes them for processing;
  - Command - executes a user command;
  - Editor data - manages the editor data structure;
  - Ancillary data - manages other data such as styles and preferences;
  - File system - manages file I/O;
  - Display - updates the screen display.

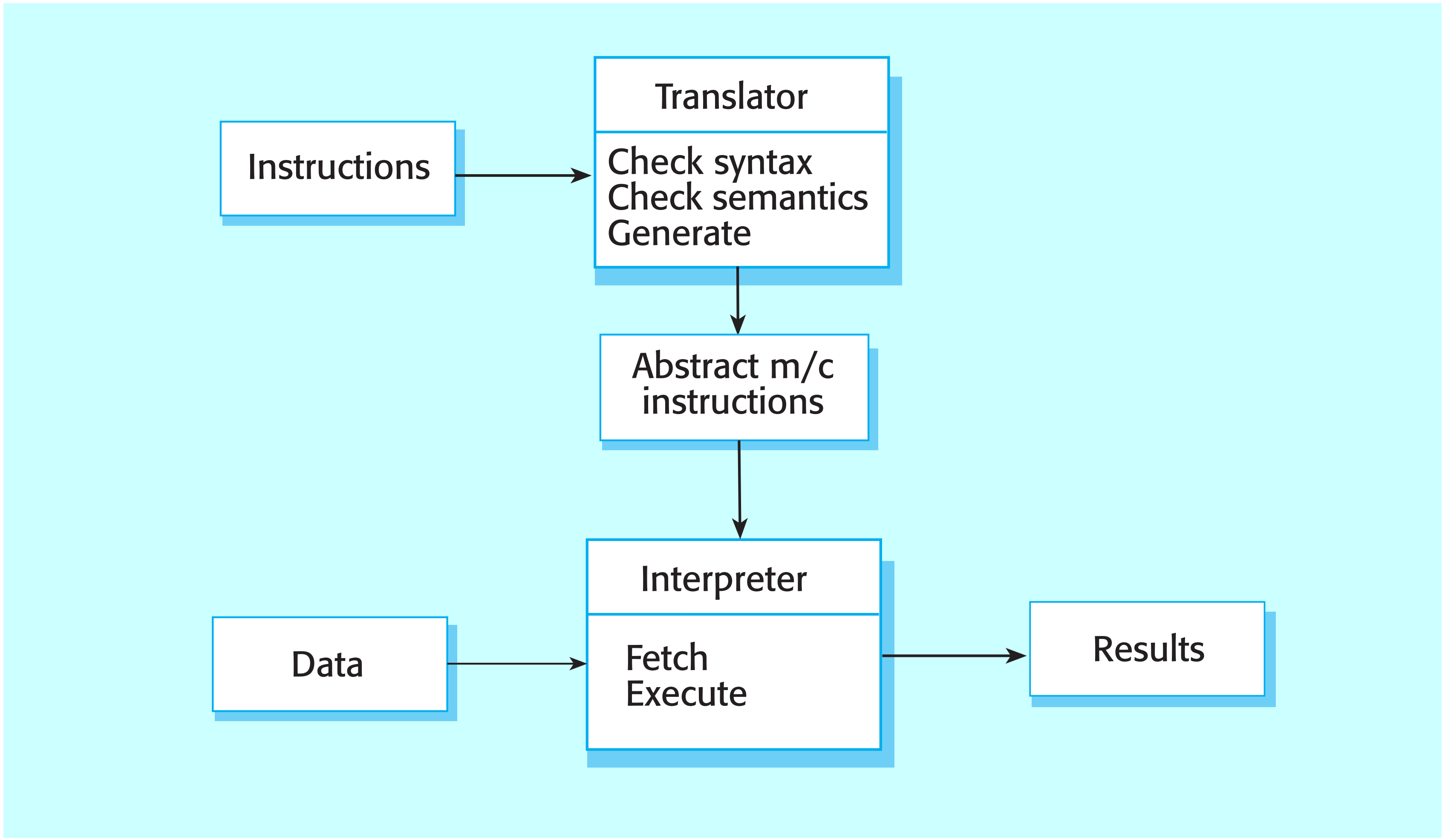
Editing system architecture



Language processing systems

- Accept a natural or artificial language as input and generate some other representation of that language.
- May include an interpreter to act on the instructions in the language that is being processed.
- Used in situations where the easiest way to solve a problem is to describe an algorithm or describe the system data
  - Meta-case tools process tool descriptions, method rules, etc and generate tools.

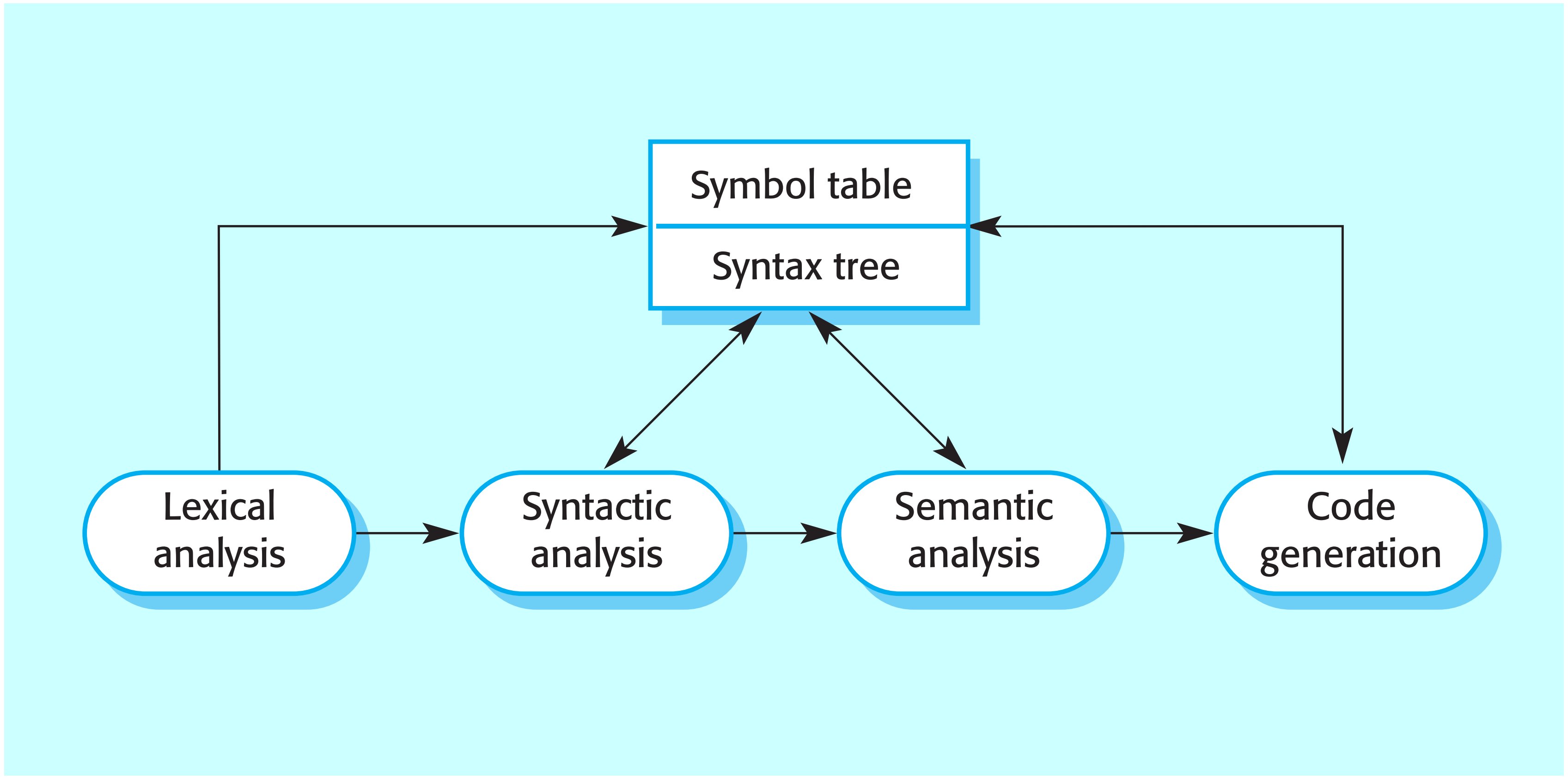
A language processing system



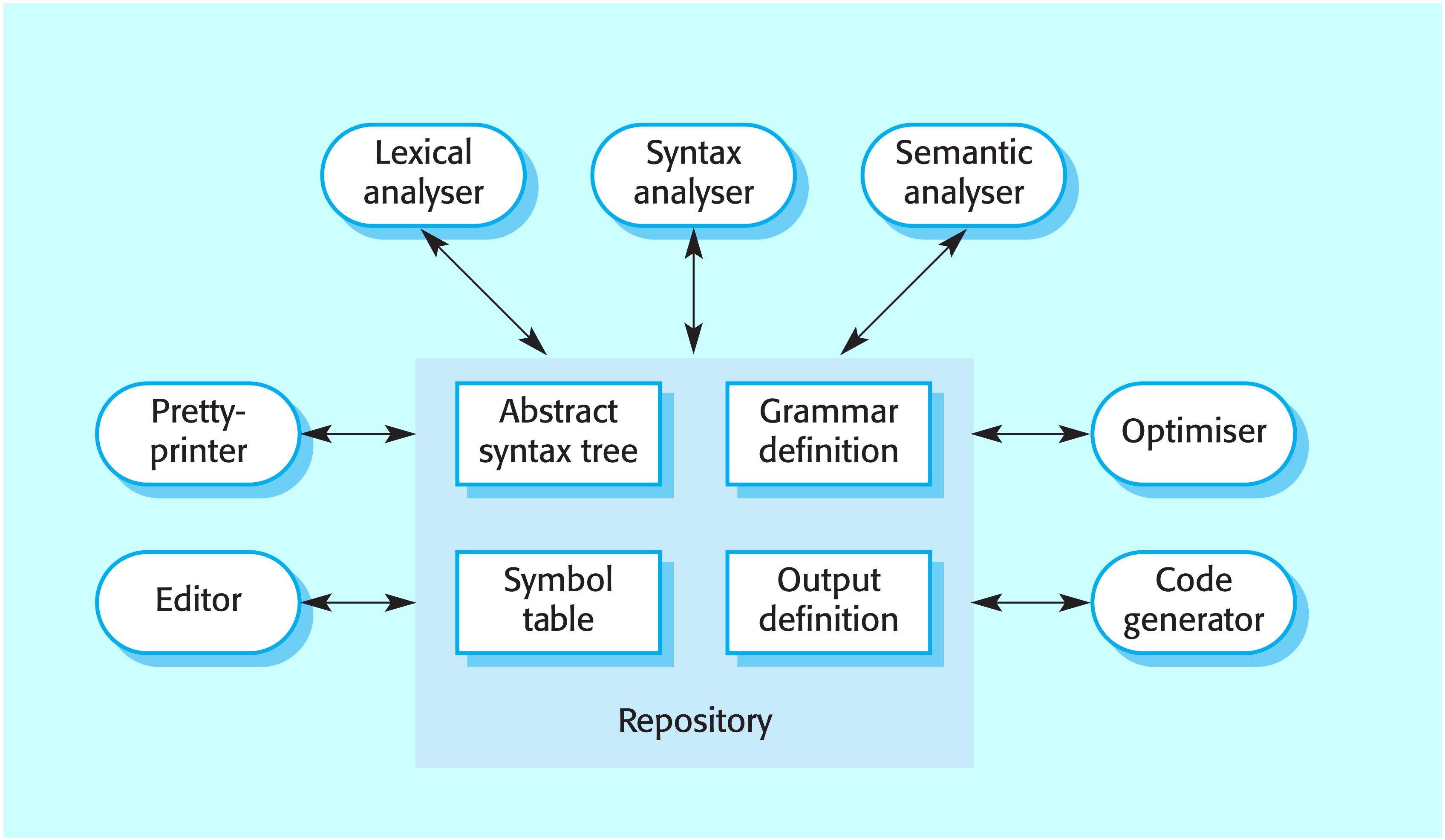
Language processing components

- Lexical analyser
- Symbol table
- Syntax analyser
- Syntax tree
- Semantic analyser
- Code generator

Data-flow model of a compiler



Repository model of a compiler


Lexical analyser
Syntax analyser
Semantic analyser
Pretty-printer
Abstract syntax tree
Grammar definition
Optimiser
Editor
Symbol table
Output definition
Code generator
Repository