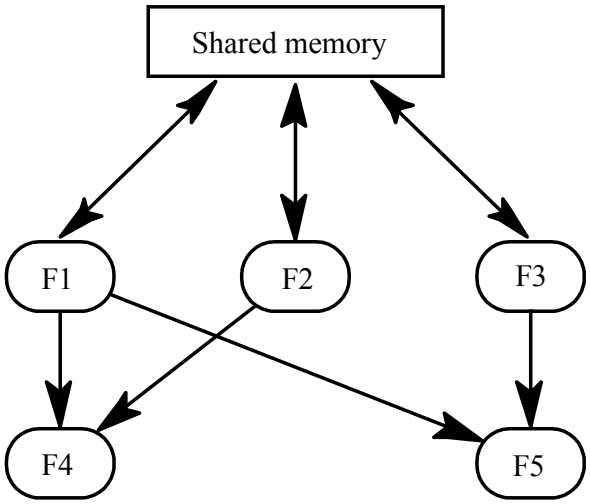
Function-oriented design

- Design with functional units which transform inputs to outputs

Function-oriented design

- Practised informally since programming began
- Thousands of systems have been developed using this approach
- Supported directly by most programming languages
- Most design methods are functional in their approach
- CASE tools are available for design support

A function-oriented view of design



Natural functional systems

- Some systems are naturally function-oriented
- Systems which maintain minimal state information, i.e. where the system is concerned with processing independent actions whose outcomes are not affected by previous actions
- Information sharing through parameter lists
- Transaction processing systems fall into this category. Each transaction is independent.

An ATM system design

- Replace with portrait slide

Functional and object-oriented design

- For many types of application, object-oriented design is likely to lead to a more reliable and maintainable system
- Some applications maintain little state -- function-oriented design is appropriate
- Standards, methods and CASE tools for functional design are well-established
- Existing systems must be maintained - function-oriented design will be practised well into the 21st century

Functional design process

- Data-flow design
  - Model the data processing in the system using data-flow diagrams
- Structural decomposition
  - Model how functions are decomposed to sub-functions using graphical structure charts
- Detailed design
  - The entities in the design and their interfaces are described in detail. These may be recorded in a data dictionary and the design expressed using a PDL (Program Description Language)

Data flow diagrams

- Show how an input data item is functionally transformed by a system into an output data item
- Are an integral part of many design methods and are supported by many CASE systems
- May be translated into either a sequential or parallel design. In a sequential design, processing elements are functions or procedures; in a parallel design, processing elements are tasks or processes

DFD notation

- Rounded rectangle - function or transform
- Rectangle - data store
- Circles - user interactions with the system
- Arrows - show direction of data flow
- keywords and/ or. Used to link data flows

Design report generator

- Replace with portrait slide

Structural decomposition

- Structural decomposition is concerned with developing a model of the design which shows the dynamic structure, i.e. function calls
- This is not the same as the static composition structure
- The aim of the designer should be to derive design units which are highly cohesive and loosely coupled
- In essence, a data flow diagram is converted to a structure chart

Decomposition guidelines

- For business applications, the top-level structure chart may have four functions namely input, process, master-file-update and output
- Data validation functions should be subordinate to an input function
- Coordination and control should be the responsibility of functions near the top of the hierarchy

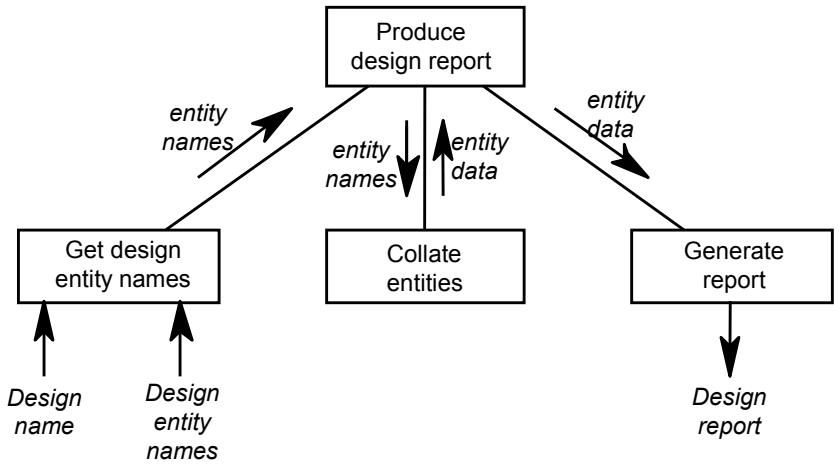
Decomposition guidelines

- The aim of the design process is to identify loosely coupled, highly cohesive functions. Each function should therefore do one thing and one thing only
- Each node in the structure chart should have between two and seven subordinates

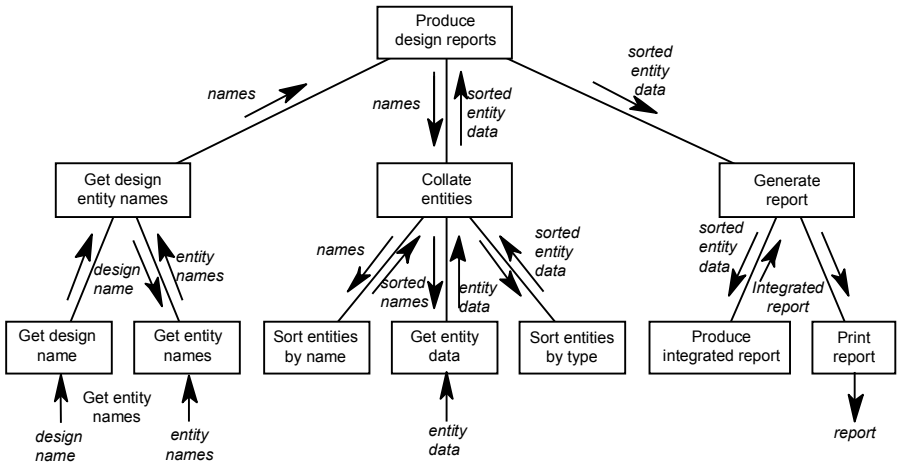
Process steps（STS分割）

- Identify system processing transformations
  - Transformations in the DFD which are concerned with processing rather than input/output activities. Group under a single function in the structure chart
- Identify input transformations
  - Transformations concerned with reading, validating and formatting inputs. Group under the input function
- Identify output transformations
  - Transformations concerned with formatting and writing output. Group under the output function

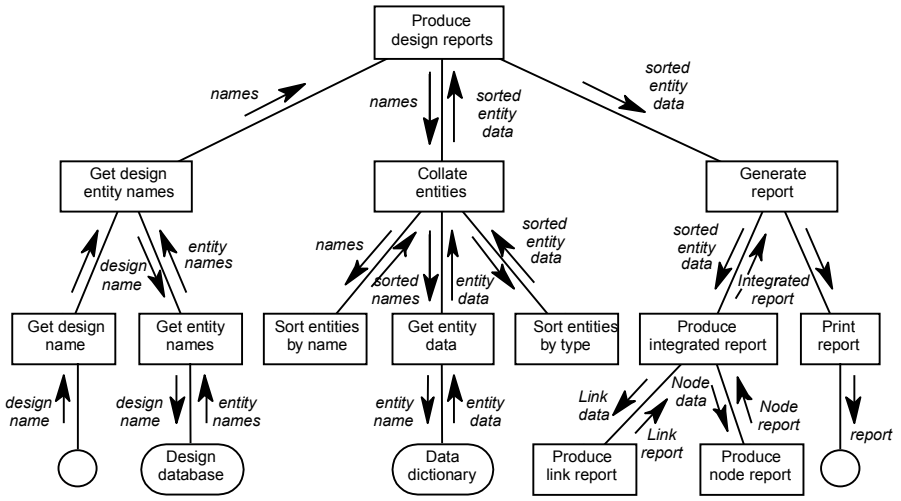
Initial structure chart



Expanded structure chart



Final structure chart



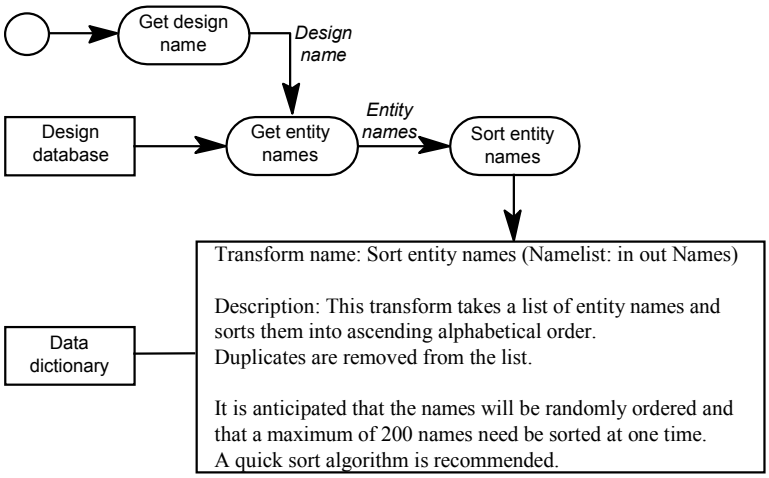
Detailed design

- Concerned with producing a short design specification (minispec) of each function. This should describe the processing, inputs and outputs
- These descriptions should be managed in a data dictionary
- From these descriptions, detailed design descriptions, expressed in a PDL or programming language, can be produced

Data dictionary entries

Entity name	Type	Description
Design name	STRING	The name of the design assigned by the design engineer.
Get design name	FUNCTION	*Input*: Design name *Function*: This function communicates with the user to get the name of a design that has been entered in the design database. *Output:* Design name
Get entity names	FUNCTION	*Input*: Design name *Function*: Given a design name, this function accesses the design database to find the names of the entities (nodes and links) in that design. *Output*: Entity names
Sorted names	ARRAY of STRING	A list of the names of the entities in a design held in ascending alphabetical order.

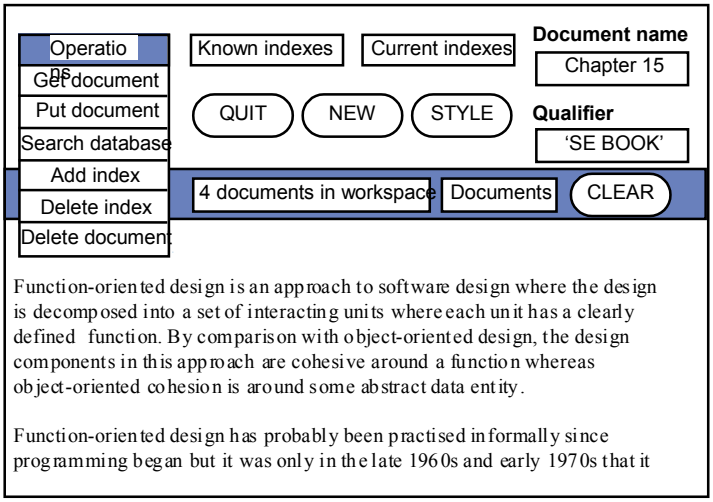
Design entity information



A comparison of design strategies

- An example of an office information retrieval system (OIRS) is used to compare different design strategies
- Functional design, concurrent systems design and object-oriented design are compared
- The OIRS is an office system for document management. Users can file, maintain and retrieve documents using it

OIRS user interface



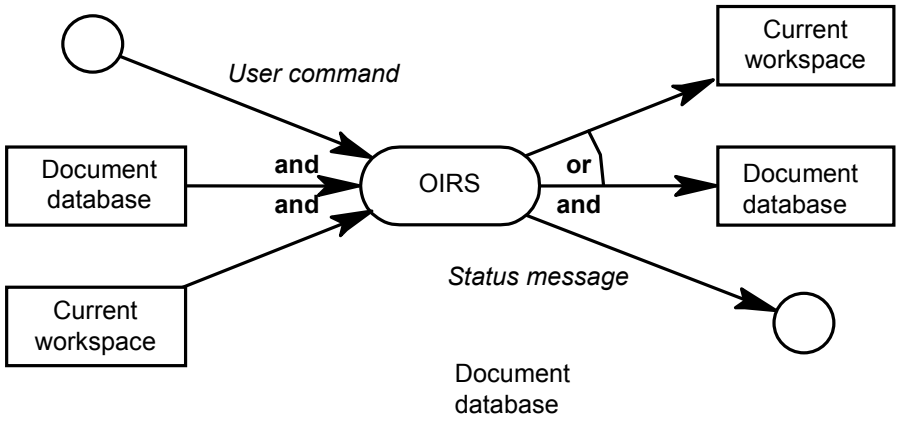
Function-oriented design is an approach to software design where the design is decomposed into a set of interacting units where each unit has a clearly defined function. By comparison with object-oriented design, the design components in this approach are cohesive around a function whereas object-oriented cohesion is around some abstract data entity.

Function-oriented design has probably been practised informally since programming began but it was only in the late 1960s and early 1970s that it

Interface description

- Operation field.
  - Pull-down menu allowing an operation to be selected.
- Known and current indexes fields
  - Pull-down menus of indexes
- Document name.
  - Name under which the document is to be filed.
- Qualifier field
  - Pattern used in retrieval.
- Current workspace
  - Contains the documents currently being used. May be edited with word processor.

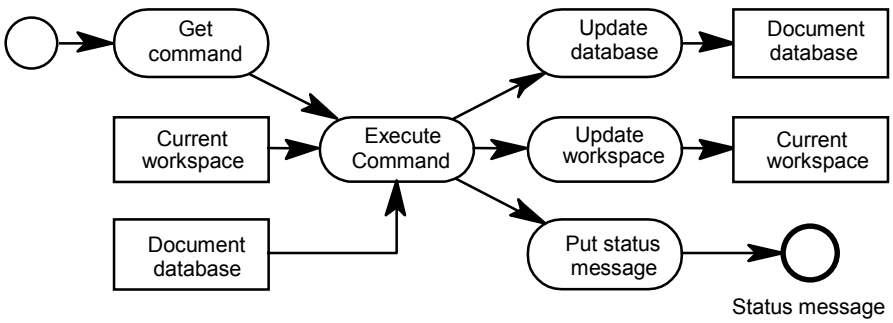
OIRS inputs and outputs



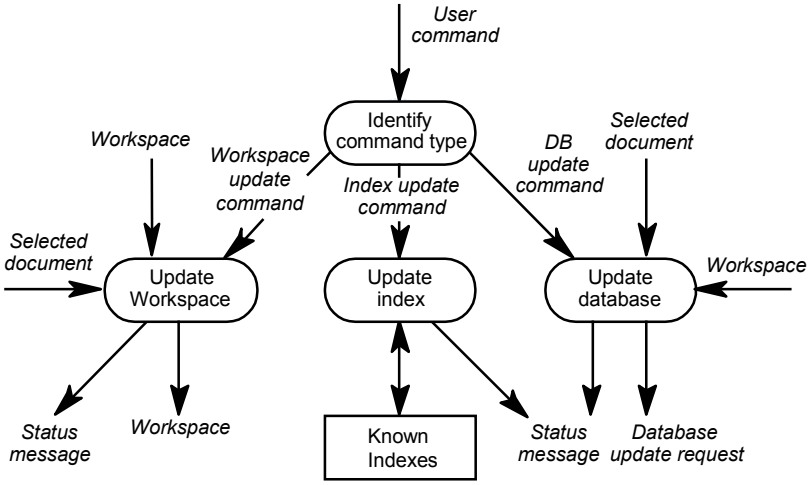
Fetch-execute model

```
procedure Interactive_system is
begin
   loop
      Command := Get_command;
      if Command = "quit" then
         -- Make sure files etc. are closed properly
         Close_down_system ;
         exit ;
      else
         Input_data := Get_input_data ;
         Execute_command (Command, Input_data, Output_data) ;
      end if ;
   end loop ;
end Interactive_system ;
```


Top-level OIRS DFD



Execute command DFD



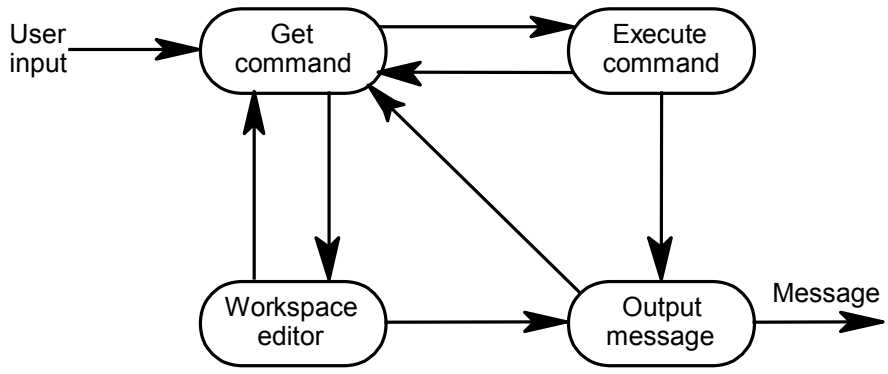
OIRS design description

- Replace with portrait slide

Concurrent systems design

- Data flow diagrams explicitly exclude control information. They can be implemented directly as concurrent processes.
- Logical groups of transformations can also be implemented as concurrent processes, e.g. input data collection and checking
- The OIRS system can be implemented as a concurrent system with command input, execution and status reporting implemented as separate tasks

OIRS process decomposition



Detailed process design

```
procedure Office_system is
    task Get_command ;
    task Process_command is
        entry Command_menu ;
        entry Display_indexes ;
        entry Edit_qualifier ;
        -- Additional entries here. One for each command
    end Process_commands ;
     task Output_message is
        entry Message_available ;
     end Output_message ;
    task Workspace_editor is
        entry Enter ;
        entry Leave ;
    end Workspace_editor ;
```

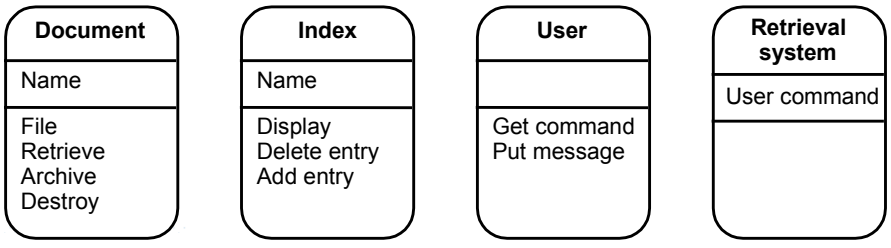

Detailed process design

- Replace with portrait slide

Object-oriented design

- An object-oriented design focuses on the entities in the system rather than the data processing activities
- Simplified OOD here which illustrates a different decomposition
- The initial decomposition was introduced in Chapter 14 in the discussion of object identification

Preliminary object identification



New objects required

- Workspace
  - Corresponds to the user's workspace and provides operations to add and remove documents from the workspace
- Index list
  - Provides facilities to manage a list of indexes
- Document database
  - Corresponds to the database of documents. provides search and retrieval operations

Additional OIRS objects

Object refinement

- Retrieval system does not provide services. It coordinates other objects. It has only attributes
- Documents and indexes are explicitly named
- The individual command components have been bundled into a single attribute User command in Retrieval system
- The User object has been replaced by the Display object

Modified OIRS objects