Data Analysis for Data Base Design

D. R. Howe

Principal Lecturer in Data Processing
Leicester Polytechnic

Edward Arn

First published 1983 by
Edward Arnold (Publishers) Ltd,
41 Bedford Square, London WC1B 3DQ

British Library Cataloguing in Publication Data

Howe, D.R.
Data analysis for data base design.
1. Data Base Management
I. Title
001.64'42 QA76. 9.D3

ISBN 0-7131-3481-X

Printed in Great Britain by
Thomson Litho Ltd, East Kilbride, Scotland.

Contents

Part A

Introduction

Part A sets the scene by using the experiences of an imaginary manufacturing company to explore the advantages and disadvantages associated with the sharing of data between applications.

A

Introduction

A.1 Data base

A data base is a collection of non-redundant data shareable between different application systems.

What does this definition mean? What is non-redundant data? Why share data? What problems arise in sharing data and how can they be overcome? We will begin to explore these questions by considering the problems encountered by a mythical manufacturing company, Torg Ltd, in the development of their computer systems.

A.2 Torg Ltd

The management of Torg Ltd, knowing that many pitfalls await the unwary in the development of computerised systems, had started cautiously by implementing a simple system for printing an up-to-date product catalogue every month. This Catalogue system maintained a master Catalogue file (Fig. A.1) comprising the data items product-number, product-description, and price. At each month-end the Marketing department updated the file to reflect price changes, the addition of new products to the catalogue and the deletion of obsolete products. The update run printed a new catalogue listing which was then reprographed for circulation to customers.

Encouraged by the success of the Catalogue system, Torg decided to try something a little more ambitious, namely a Stock control system for the Stores department. The data required for this system would be product-number, product-description, quantity-in-stock, and re-order-level. The quantity-in-stock of each product would be updated weekly with stock movement data, and an exception report would be printed showing those products for which the quantity-in-stock had fallen below the re-order-level.

Geoff Watson, the data processing manager, agreed with his chief (and only) systems analyst Tom Cross that since much of the data required by the Stock system was already held on the Catalogue system (viz. product-number, product-description), it would be sensible to use the same master file for both systems. The Catalogue file was therefore extended to include quantity-in-stock and re-order-level data items and was renamed the Product file (Fig. A.2). Programs were written for the Stock system to handle stock movements and changes to re-order-levels. The update program in the Catalogue system had to be amended to cope with the additional data items in the Product file, but this was considered to be a trivial change. As illustrated in the diagram, the Stores department was responsible for the weekly updating of quantities-in-stock and for the revision of re-order-levels when necessary. The Marketing department continued to be

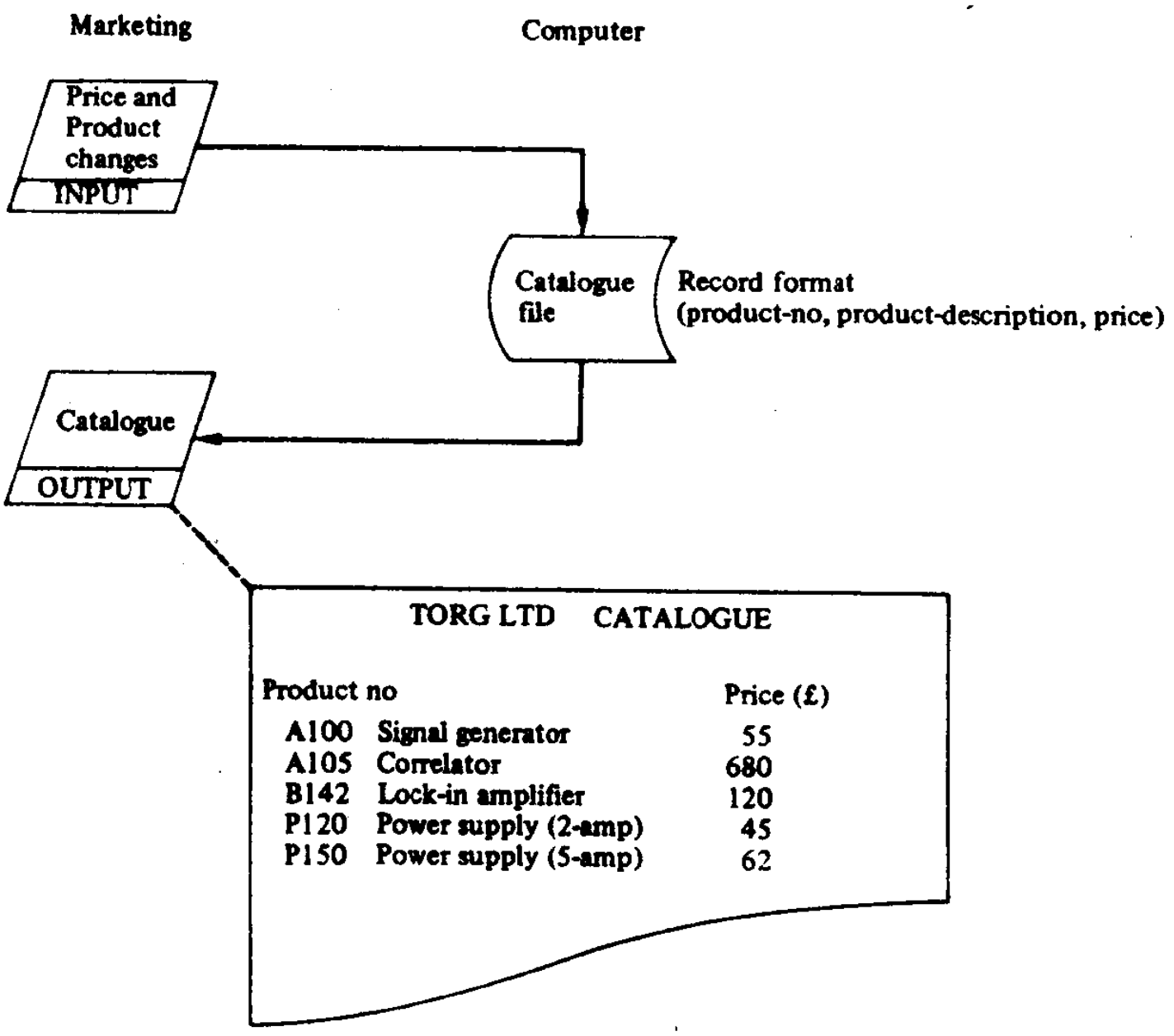

Fig. A.1 The Catalogue system

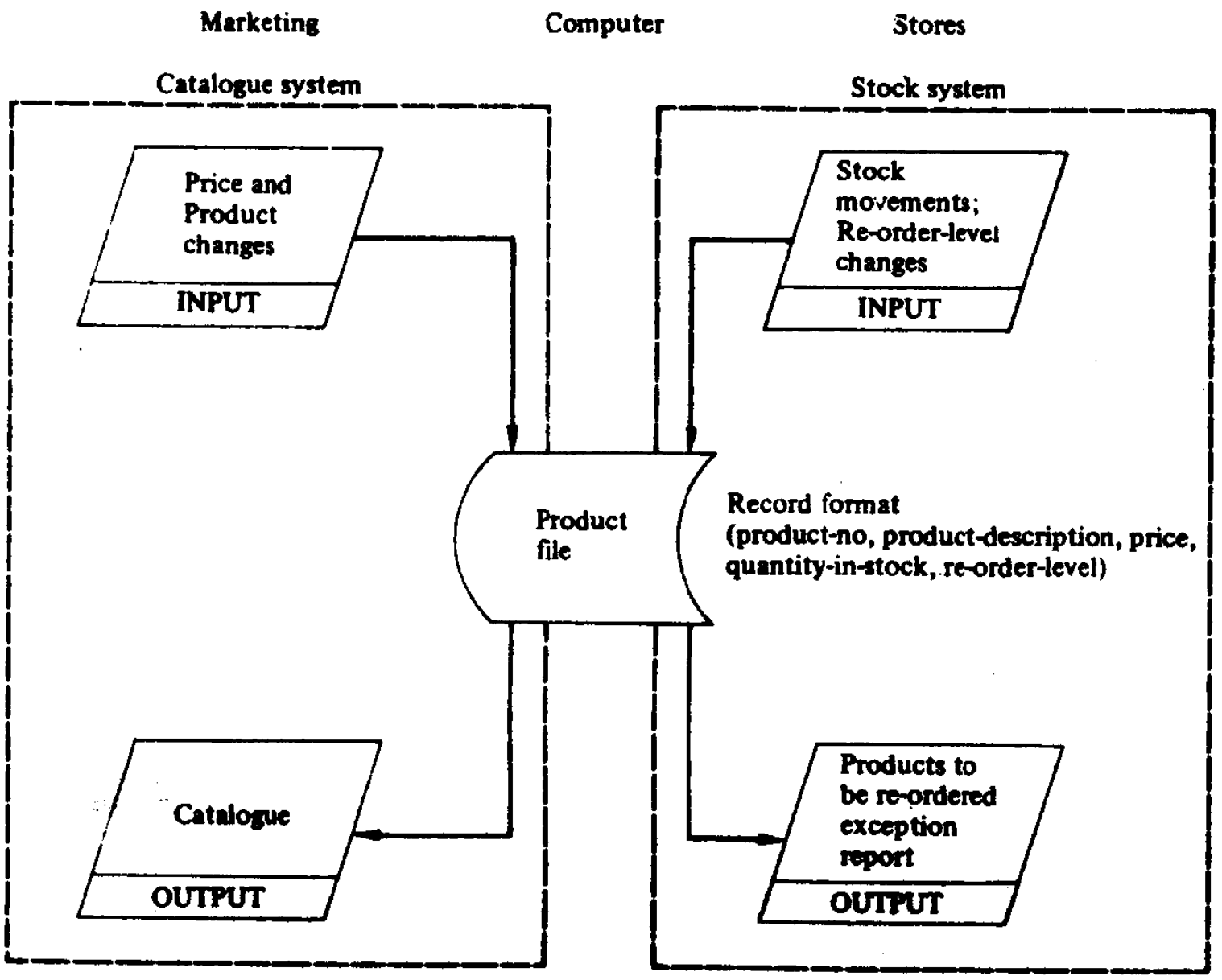

Fig. A.2 Joint Product file shared by Catalogue and Stock systems

The Stock system was thoroughly tested and put into operation at the beginning of September. Early in October disaster struck. The latest product catalogue (Fig. A.3) had somehow acquired some additional columns of figures, namely the values of quantity-in-stock and re-order-level. Unfortunately, the error was not noticed until after the new catalogues had been distributed to customers. Several customers concluded that a company which could not print an intelligible catalogue was best avoided. A competitor discovered the meaning of the figures simply by telephoning one of the Torg secretaries. As a result the competitor identified several products for which Torg was having difficulty obtaining supplies and, by changing its marketing strategy, the competitor was able to win over some important Torg customers.

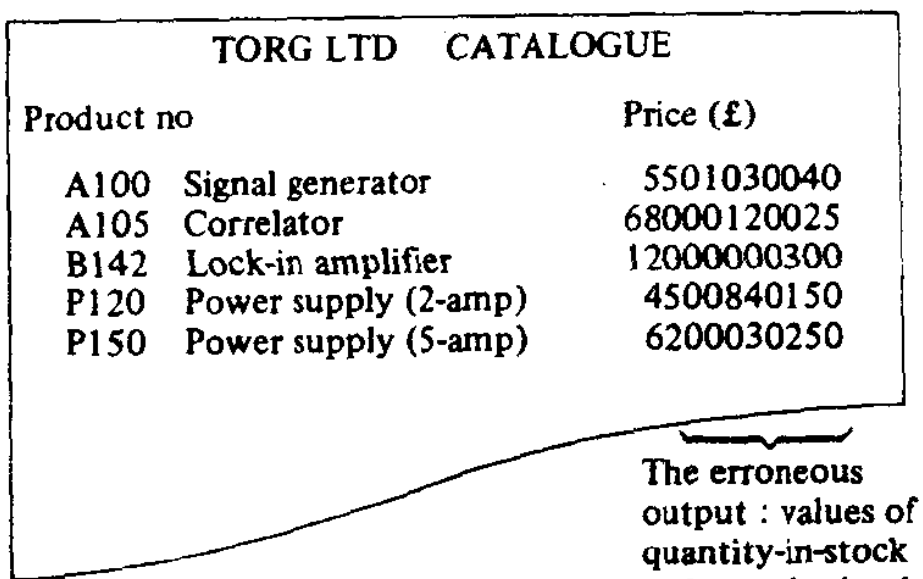

Fig. A.3 The corrupted catalogue

An error in the Catalogue printing program was soon traced and corrected but the 'Catalogue Catastrophe', as it became popularly known within the company, prompted Watson to rethink the systems development policy. In future, he decided, each of the company's systems would be designed around its own master files. Not only would this application-centred approach eliminate the possibility of data being seen by an unauthorised program, but it would also make it possible to concentrate on the development of one application at a time.

To start the good work the Catalogue and Stock systems were split. The Catalogue system reverted to its original design and a new Stock system was designed around its own Stock file (Fig. A.4). The Product Design Department, who authorised the addition and deletion of products, were responsible for notifying both Marketing and Stores of such changes. This solution worked successfully at first, but after some while complaints were made by the Sales manager that the descriptions of products in the stock report were sometimes inconsistent with those in the catalogue. There were also instances of a product appearing in the catalogue but not in the stock report (and *vice versa*).

Investigation showed that there were several reasons for the discrepancies. One problem was that Stores ran its update program weekly, whereas Marketing ran its update program monthly. Another problem was that mistakes by Product Design meant that Marketing and Stores did not always receive the same product change data. Occasionally data was accidentally omitted from an update run, and in a few cases errors had been made by the computer operators.

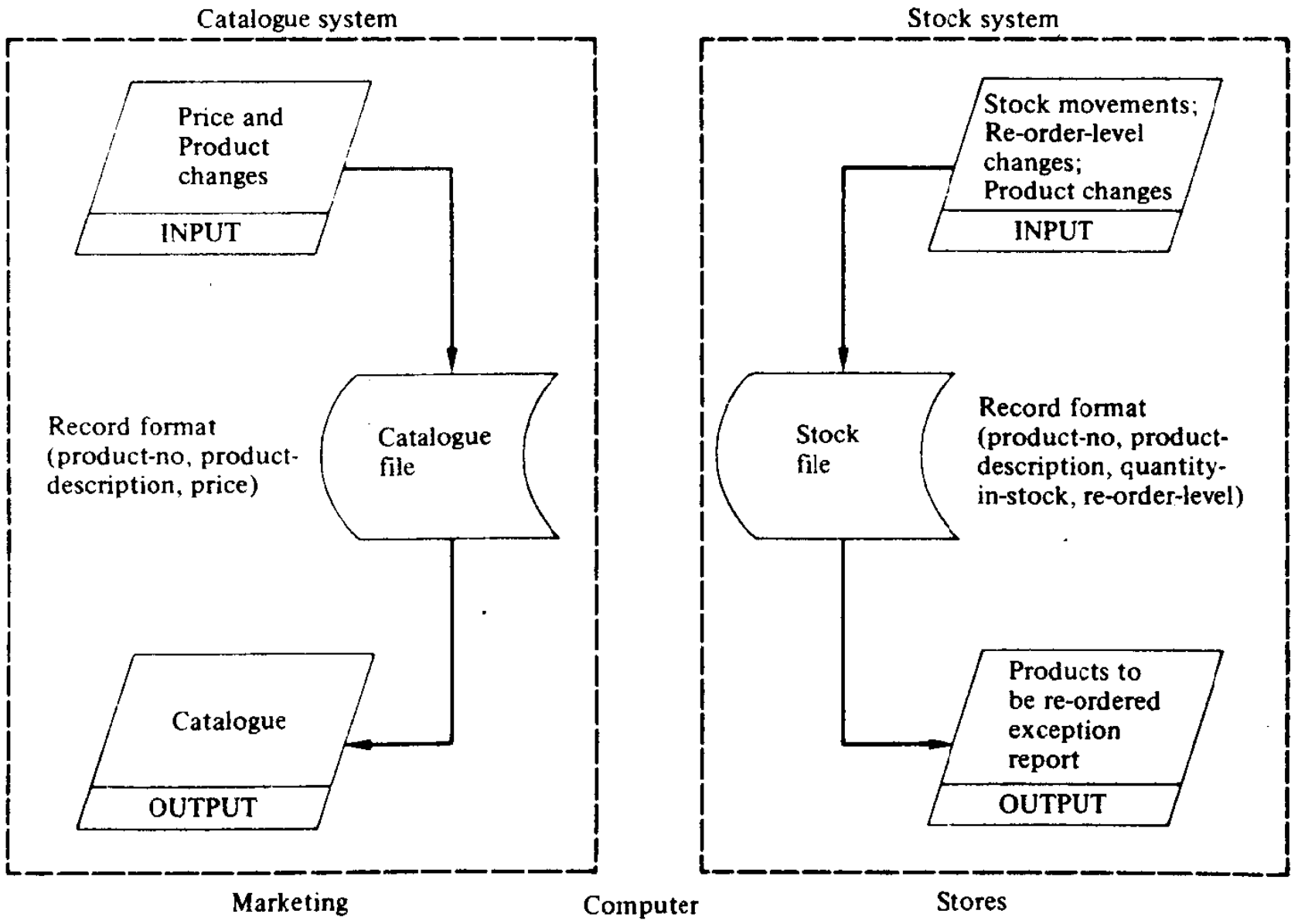

Fig. A.4 Application-centred solution. Separate Catalogue and Stock systems

almost inevitably lead to inconsistent data. After several rather acrimonious meetings, in which each of the parties blamed everyone else, it was agreed that the data processing department would write a 'consistency audit' program which would compare the two files and print a list of discrepancies. This program would be run every few months so that discrepancies could be identified and corrected. Furthermore, memos were sent to the clerical staff in Product Design, Marketing and Stores, emphasising the need for care in dealing with product changes.

During the next few years several other systems were implemented, including Production Scheduling, Purchasing, Payroll, and Customer Orders. The policy of developing each system in isolation had been continued, in the sense that each system had its own master files, but the growing need to communicate data between systems had led to the use of transfer files. For example, information about products ordered by customers was passed from the Customer Orders system to the Production Scheduling system via a transfer file. As the complexity of the systems grew, the number of transfer files proliferated (Fig. A.5).

The problem of consistency between systems was by now a real headache. Production Scheduling, Customer Orders, Purchasing, Catalogue and Stock Control systems all contained data on products, and much of the data was inconsistent. The consistency audit program technique was still used, but it had become practically impossible to maintain consistency in the face of differing update cycles and data input errors. The proliferation of transfer files caused problems in scheduling work on the computer. A further worry for Watson was the waste of storage entailed by the duplication of data between files. Watson knew that if the duplication of data were eliminated, more master files could be held on-line simultaneously and better use made of the computer's multiprogramming capabilities.

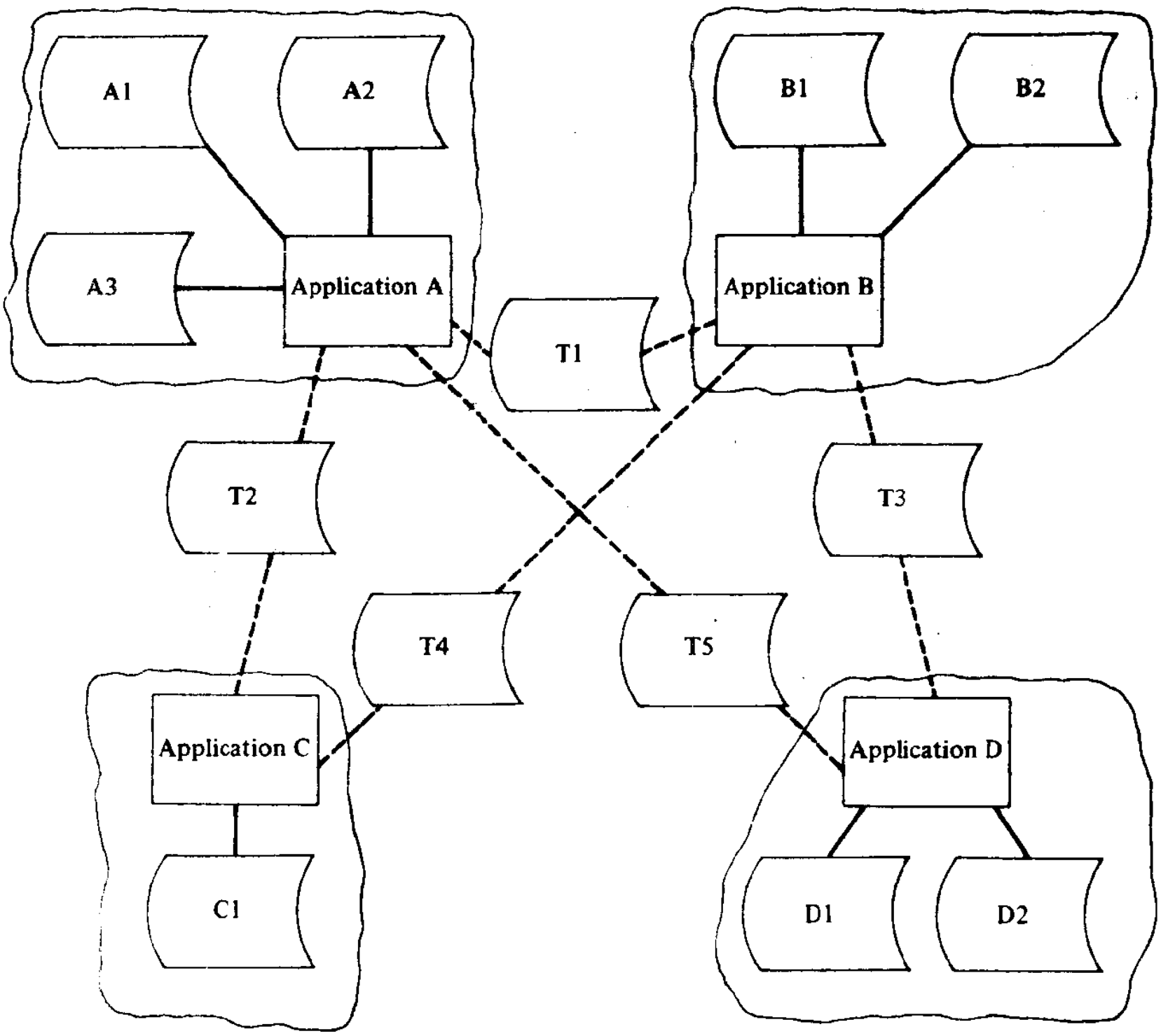

Fig. A.5 Proliferation of transfer files (T1–T5) to communicate between application systems

By this time *unproductive maintenance*, as Watson called it, was occupying a substantial amount of the programmers' time. The problem was well illustrated by the Payroll system for which some fifteen *ad hoc* report programs had been developed. These programs were not part of the main payroll suite but were run on request to provide management information. A new agreement between management and the Sports and Social Club that membership fees would be deducted automatically from members' pay meant that it was necessary to distinguish in the Payroll record between members and non-members. An extra data item was therefore added to the Payroll record so that club membership could be indicated. Certain Payroll programs which made use of the new data item naturally had to be amended, and this work Watson regarded as productive maintenance. The unproductive maintenance involved the amendment of the other Payroll programs, including all fifteen *ad hoc* report programs, which made no use of the new data item, but which had to be amended (in order to make their record descriptions consistent with the new record size) and then re-compiled and tested. Watson recognised that the root of the problem lay in the sharing of the same Payroll file by many programs, but that it was even less of a practical proposition to

computer files was a valuable resource which could be tapped to provide management information reports to supplement those provided by the standard application systems. A common complaint voiced by management was that the data processing department took weeks, or even months, to write a report program. Consequently, a report was often not available to a manager until too late to be of any use. The data processing department replied that many report programs, which had been justified on the grounds that they would be used frequently, had been used once only, and that management should know better than to waste the time of an overstretched department.

Up to this point, all Torg's systems had been run in batch mode. Watson decided that the time had come to take the plunge into on-line terminal systems. It was decided that the Catalogue system would be converted to on-line updating and retrieval, with the primary aim of better servicing of telephoned enquiries from customers. Marketing asked for on-line access to the Catalogue file via either a product-number or a product-description. Unfortunately, none of the standard file organisation routines available within the data processing department could provide a sufficiently fast response to both types of access. A serial file[1] was plainly out of the question. A random file[2] with product-number as the key would have provided fast access by product-number, but access by product-description would have entailed a serial search. The same objection applied to an indexed sequential file.[3] The basic problem was that none of the file organisations permitted direct access via more than one key.[4] Eventually, a solution was reached in which only product-number could be used for on-line access; retrieval by product-description was dealt with by referring to a printed list produced in the monthly batch run.

With the implementation of the company's first on-line terminal system successfully completed, Watson reviewed the state of computer systems development. The old policy of application-centred development had to all intents and purposes broken down. The spread of computer systems throughout the company had reached the point at which no major new system could be implemented without considering its interaction with existing computer systems. In spite of the experience gained over the years by the data processing department it seemed to be becoming harder rather than easier to cope with systems development. Perhaps a new man would be able to sort things out. Watson handed in his resignation and set up as an independent consultant.

A.3 Analysis

The problems experienced by Torg Ltd can be summarised as follows.

(1) The duplication of data between different master files is a potential source of inconsistent data.
(2) The sharing of data between systems poses a security problem.

[1]As typified by a magnetic tape file.
[2]i.e. where a randomising algorithm is used to calculate a record's address from its key.
[3]i.e. where an index of record keys allows efficient access both in key sequence and directly to individual records.
[4]This is true of the conventional file organisation routines available at Torg Ltd, but not of the more sophisticated implementations discussed in Part 4.

(3) Unproductive maintenance can become a serious drain on the resources of a data processing department.
(4) Managers cannot exploit fully the data available to the computer because of delays in getting report programs written.
(5) The greater the number of existing computer systems, the longer it is likely to take to develop new systems, because there are more interactions to consider.
(6) File organisations supporting only single-key direct access may be inadequate, particularly for on-line systems.
(7) Systems were developed rather haphazardly. There was no coordinated view of the data as a whole, nor was there a proper systems development plan.

In the next chapter we contrast Torg's application-centred approach to systems design with a *data base* approach and, in doing so, introduce the idea of a *data base management system*.

Questions

1. Could the 'Catalogue Catastrophe' (section A.2) have been avoided by better program testing? What are the implications of your answer if a large number of application programs share the same file? (2 min)

2. The catalogue printing program worked correctly with a record format of (product-no, product-description, price) as in Fig. A.1 but produced incorrect output (Fig. A.3) when its record description was altered to match the larger record size of the Product file in Fig. A.2. Assuming that the program was written in COBOL and that the only change was to add a FILLER data item to the record description, suggest a cause for the incorrect output. Was there a bug in the original version of the program or not? Do you have any criticisms to make concerning the program or record design? (5 min)

3. Assuming that the Catalogue system and Stock system share a data base as in Fig. A.2, discuss whether there is likely to be conflict over the ownership of data (i.e. the right to amend, add, delete, or restrict access to the data). (5 min)

4. The Product file in Fig. A.2 contains no redundant data. Does it follow that the file necessarily contains mutually consistent data? (2 min)

5. Which data items are common to both the Catalogue file and the Stock file in the application-centred solution of Fig. A.4? Are the data items common to both files the only source of inconsistency? (2 min)

6. Is better clerical procedures design the answer to inconsistent updating of duplicated data? (2 min)

7. Suppose duplication of data between application systems is allowed, but inconsistency is to be avoided by updating all copies of duplicated data 'simultaneously'. What are the implications of this procedure with regard to (a) batch systems, (b) on-line terminal systems, (c) recovery from computer malfunction? (10 min)

8. The use of transfer files (Fig. A.5) might be a security risk. Does the use of transfer files have any advantages over allowing applications to access each other's master files directly? (2 min)

9. Why do you think 'The proliferation of transfer files caused problems in scheduling

10. Do you think the use of consistency audit programs is a good idea, or is it just papering over the cracks in a basically unsound approach? (3 min)

Answer pointers

1. Yes. But with a large number of programs to test the chance of faulty testing is increased. (Is the relationship linear?)

2. One possible explanation is that the print-record-area was defined as being the same size as the printer page width (say, 132 characters) and that the input record was moved to the print-record-area by a group MOVE of the record as a whole. In each version of the program, the excess characters in the print-record-area would be padded automatically with spaces, but in each case the whole of the input-record would be transferred to the print-record-area.

There was not a bug in the original version of the program unless the original specification included maintainability criteria which were not met. The problem would not have occurred if item-by-item moves had been used instead of a group move. The record design is unusual in that the product-description values as stored on the Product file are already surrounded by the formatting blanks required to separate them from the product-no and price values on the printed output (otherwise a group move would not work). One might also wonder if it is wise to restrict prices to whole numbers of pounds.

3. The Stores department might object to Marketing deleting the record for a product which has been withdrawn from the catalogue, but for which Stores still have supplies in stock which need to be scrapped.

4. No. For example, an input error might mean that a product-description and product-no are inconsistent. The fact that there are two sources of updates to the Product file may also lead to inconsistency. Suppose that, because of an increase in the price of a product, it has been decided that the re-order-level should be reduced. If the updates performed by the Catalogue and Stock systems are not synchronised the relevant product record will temporarily contain inconsistent values for price and re-order-level. A report derived from this file while it is in an inconsistent state would convey misleading information if it included the values of price and re-order-level. Similarly, the two reports shown in Fig. A.2 could be inconsistent.

5. Product-no, product-description. No. See answer pointers for question 4.

6. It may help, but cannot guarantee consistency.

7. Some implications are: (a) all copies of the duplicated data must be available to the updating program when it is run, which may mean the mounting of a large number of files; (b) the response time to terminal users may be unacceptably long because of the time taken to update all copies of the data; (c) the recovery procedure may be complicated by the larger number of files involved.

8. Transfer files will usually have smaller records and give better security because only those data items to be transferred should be written to them; they may contain fewer records; they may have a more convenient file organisation; it is simpler to audit the contents of transfer files than to audit the operation of a program which accesses master files directly.

9. The need to mount several transfer files (both input and output files) simultaneously. The need to ensure that the input transfer files are at a consistent level of update.

10. As used by Torg, a case of papering over the cracks. We shall see later that even the best designed data base is likely to contain some duplicated data, in which case the use of consistency audit programs is sensible, but as a check that all is well rather than as a firefighting exercise.