Neutral Citation Number: [2004] EWHC 1725 (Ch)

Case No: HC02 C01268

IN THE HIGH COURT OF JUSTICE
CHANCERY DIVISION

Royal Courts of Justice
Strand, London, WC2A 2LL

Date: 30 July 2004

Before :

THE HONOURABLE MR JUSTICE PUMFREY

- -

Between :

NAVITAIRE INC	**Claimant**
- and -	
(1) EASYJET AIRLINE COMPANY	
(2) BULLETPROOF TECHNOLOGIES INC	**Defendants**

- -
- -

Henry Carr QC, Mark Vanhegan and Anna Edwards-Stuart (instructed by **Field Fisher Waterhouse**) for the **Claimants**
Richard Arnold QC and Brian Nicholson (instructed by **Herbert Smith**) for the **Defendants**

Hearing dates: 4-7, 10-14, 17-21, 24-28 November 2003, 2-6, 9-13 February 2004

- -

Redacted Approved Judgment

This judgment is the redacted version of a judgment handed down on 30 July 2004. The redactions are made at the request of the parties, to protect what is said to be confidential information relating to their respective software systems. While the parties' requests have been incorporated in this document, it does not reflect any determination that the redacted matters are in fact confidential.

....

MR JUSTICE PUMFREY

Mr Justice Pumfrey :

Introduction

1. This is an action for infringement of copyright in computer software. It concerns the software implementing an airline booking system principally employed by low-cost airlines who employ 'ticketless' booking. The claimant's system is called OpenRes. It (and its successor system) have been commercially successful, and are employed by a number of airlines. The claimant ('Navitaire') is now the owner of the copyright in the various works that go to make up the source code of OpenRes. The first defendant ('easyJet') is the well-known low-cost airline. The second defendant ('BulletProof') is a software developer located in California, who is responsible for writing the code of the allegedly infringing system, which is called eRes, in consultation with easyJet's IT department. I will refer to the claimant and its predecessors in title (Open Skies, Inc. and the Open Skies Division of Hewlett Packard) as 'Navitaire' for convenience.

2. The case is factually and technically complex and has taken a considerable amount of time in court. It raises starkly an issue of considerable importance in the law of copyright. While there are comparatively minor allegations of infringement by copying of certain code and an allegation in relation to the databases that I summarise below the striking feature of this action is that Navitaire does not suggest that easyJet or BulletProof ever had access to the source code of the OpenRes system. What is alleged, and not disputed, is that easyJet wanted a new system that was substantially indistinguishable from the OpenRes system, as easyJet used it, in respect of its 'user interface'. This term is used to denote the appearance the running software presents to the user, who may be an agent in a call centre or a private individual seeking to make a booking by use of the World Wide Web. It substantially achieved this far from simple goal. It is not in dispute that none of the underlying software in any way resembles that of OpenRes, save that it acts upon identical or very similar inputs and produces very similar results, but it is said that the copyright in OpenRes is infringed by what was called 'non-textual copying.'

3. In its context in this action, 'non-textual copying' had three aspects. The first was the adoption of the 'look and feel' of the running OpenRes software. The second, not always clearly distinguished from the first during the trial, was a detailed copying of many of the individual commands entered by the user to achieve particular results. The third was the copying of certain of the results, in the form of screen displays and of 'reports' displayed on the screen in response to prescribed instructions. In other words, as used by easyJet the systems are very similar in use. Internally, it is correct to say that they are completely different, subject to a point on the names used to identify certain data in the databases in eRes. Given that near-identity in appearance and function could not have been achieved without a close analysis of the OpenRes system in action, Navitaire say that there is here 'non-textual' reproduction of either the whole of the OpenRes software considered as a single copyright work or alternatively of the various copyrights subsisting in 'modules' going to make up the system.

4. The commands alleged to have been copied by easyJet amount to some 44% of the OpenRes commands, on the estimate of Dr Hunt, Navitaire's expert witness.

> "All of the OpenRes complex commands and nearly all of their sub-options are reproduced in eRes. 44% of the OpenRes

> simple commands are also reproduced. Most of the commands that are not reproduced, or where there are differences in sub-options, result from the fact that easyJet only used a sub-set of the OpenRes commands. For example, commands relating to standby passengers, seat assignments and connecting flights have not been reproduced, presumably because easyJet does not need these facilities."

I should make it clear at this stage that this conclusion is slightly inaccurate. Parts only of some of the complex commands are present in OpenRes because the easyJet business model has no place for certain of the options that were accordingly not incorporated; and a certain amount depends upon the sustainability of the distinction between simple and complex commands, which is discussed below.

5. There is here an issue of general importance. To emulate the action of a piece of software by the writing of other software that has no internal similarity to the first but is deliberately designed to 'look' the same and achieve the same results is far from uncommon. If Navitaire are right in their most far-reaching submission, much of such work may amount to the infringement of copyright in the original computer program, even if the alleged infringer had no access to the source code for it and did not investigate or decompile the executable program.

6. The World Wide Web interface of the OpenRes system was provided by a software module called TakeFlight. For reasons that I explain below, the TakeFlight module consists only of source code (it was written in an interpreted rather than a compiled language) and it was copied and modified on a number of occasions by easyJet. The purpose of this copying was to fix bugs, provide for the display of promotions and the like and to provide foreign language interfaces because the code was not internationalised. This copying is said to be a breach of the terms of the licence to use the software granted by Navitaire or its predecessor in title to easyJet. easyJet's own WWW interface for eRes was written in-house by easyJet's employees and, again, it cannot be suggested that the code itself was copied. Again the allegation is of 'non-textual copying' of the software by producing a user interface having the same 'look and feel' as TakeFlight.

7. An airline booking system depends crucially upon the underlying method of recording data relating to availability of flights, availability of seats, details of passengers, their flights, their baggage, and so on. This data is what is called 'persistent' data and must be recorded in databases. The principal function of the airline booking system software is to read data from the relevant databases in response to requests from the user and to record details specified by the user, while adjusting the former in dependence upon the latter. The allegation of infringement of copyright in respect of the databases had two aspects. The first was that in transferring or 'migrating' the data contained in their OpenRes databases, which contained a record of every passenger and every flight on an easyJet aeroplane, to the new system, easyJet made interim copies of the existing OpenRes databases that they were not entitled to make. The second is that easyJet and BulletProof used their knowledge of the OpenRes databases to design the eRes databases in such a way that the copyright or copyrights alleged to subsist in certain 'schemas' which define aspects of the structure of the OpenRes database have been reproduced in the structure of the eRes database.

8. Factually the claim in respect of the user interfaces of the eRes system does not present serious difficulty. There is a dispute as to originality, or more accurately to the amount of originality in the works relied on, but the problems are essentially legal ones once the technical aspects have been understood. The position in respect of the databases is different. There is a serious dispute as to the material employed by BulletProof in specifying the eRes database structures, and whether and to what extent information about the OpenRes databases gleaned during the data migration process was used, and, if it was used, whether that use was illegitimate. There is again no doubt that none of the code used to manipulate the databases themselves can be said to have been copied, and for reasons that I explain below there is now no suggestion that the databases in eRes are in some way manipulated in a manner similar to that in OpenRes.

9. Procedurally, there has been one important problem with the case. By the order of Master Moncaster made 18 February 2003, Navitaire were required to provide complete particulars of each and every similarity between OpenRes and eRes relied upon in support of its allegations of infringement. Rather than supply such particulars, Navitaire's advisers required Navitaire's expert witness, Dr Gillian Hunt, to prepare an expert report that concerned itself only with the similarities between the systems. It said nothing about the differences and it did not provide alternative explanations for features alleged to be copied. It was in effect the indictment. This was not compliance with the Master's order, and it was not fair to Dr Hunt, who prepared an entirely one-sided document, to the extent even that if a help message contained some text similar and some text different from that in the Navitaire system, the similar part only was referred to. The report turned out to be an embarrassment, as it contained material that was not placed in context by any other report (Dr Hunt ended up making seven reports), and its nature ensured that it contained no nuance.

10. There is a great amount of technical detail in the case. Some of the issues cannot be properly approached without some understanding of the technical issues. I have tried not to overburden this judgment with technical discussion, but some reference to the source software is essential. I explain the matters relating to the user interface in my own words. The material relating to databases was more difficult, and the parties provided me with extensive extracts from two text books and the manual for TurboIMAGE XL, the database management system used in the OpenRes system. I will not attempt to repeat that information here, but my understanding of the database case has been largely informed by these publications.

11. The copyright works relied on form part of the source code of version 5.58 of OpenRes. The whole of the source code of the entire system is also relied on. It is a fairly large software system, a simple line count showing a total of about 786,000 lines of code, largely in the COBOL language, in the source files for the OpenRes 5.58 system supplied to me.

The supply of OpenRes to easyJet

12. The OpenRes system is an airline booking system that is intended for use in 'ticketless' transactions. The passenger does not receive a ticket as a result of a booking transaction, but is instead given a single reference number. The reference number is used for the purpose of check-in at the airport. A paperless model, in which the financial functions are also handled without paper and in which the structure of reservations is comparatively simple is of particular interest to low cost airlines.

13. So far as the evidence in this case is concerned, ticketless reservations were first used by Morris Air, a low-cost airline in the United States. Its founder and President, Mr Neeleman, gave evidence that he sold Morris Air to Southwest Airlines having first employed David Evans as Vice President of Information Services. Between January and June 1993, Mr David Evans wrote a reservation system called MARS (Morris Air Reservation System). This system was written for a Hewlett Packard ("HP") computer, the HP 3000, in a language called Business Basic/XL. This appears to have been a version of the BASIC language proprietary to HP. The data was stored in a database managed by the HP TurboImage system.

14. When Morris Air was purchased by Southwest Airlines in 1994, I infer from Mr David Evans's evidence that he was dismissed, either at that stage or some short time later. Notwithstanding his departure from Southwest Airlines, he developed a ticketless system comprising a data repository that derived its data from the SABRE reservation system used by Southwest. SABRE stands for Semi-Automated Business Research Environment. It was created by American Airlines in 1964, and has been developed continuously since then, now being the property of a company called Sabre Inc. It is a computer reservation system intended for use with tickets and is widely used. Mr Evan's ticketless system for Southwest Airlines was released in August 1994.

15. In April 1994 Mr David Evans signed a contract with Southwest Airlines to help them in writing a new reservation system to replace SABRE. In June 1994, he started a company called Evans Airline Information Systems Inc. He signed a contract to become a Hewlett Packard Value Added Reseller (VAR) at this time. The replacement for SABRE appears to have been written in the two years from January 1995 to December 1996. But before this work started, and while he was working on the ticketless system for Southwest Airlines, Mr David Evans was approached by two representatives of a small charter airline operating in Brussels that was then called Eurobelgian Airlines but became Virgin Express. Eurobelgian Airlines wanted to purchase MARS but, when they were told that it was the property of Southwest Airlines, agreed with Mr Evans that he would write them a ticketless booking system. This he did, on an HP 3000 purchased by Eurobelgian Airlines, in the evenings while he was working on the Southwest ticketless system. It became known as the Evans Airline Reservation System (EARS) and was completed, so I was told, in about three months, working in the evenings and at weekends. EARS is the basis of OpenRes.

16. EARS differed from MARS in that it was written in a different language (COBOL rather than BASIC) and to the specific requirements of Eurobelgian Airlines. It was only ever supplied to Eurobelgian Airlines and CityBird, another small Belgian airline.

17. In September 1995, a further enquiry from a small airline, this time WestJet from Calgary, Canada led to the employment of Greg McDaniel, a veteran of Morris Air who also gave evidence, who started to convert EARS to something suitable for WestJet in September 1995. The result of this development was OpenRes. At about this time, EAIS was purchased by David Neeleman who renamed it Open Skies, Inc. Hewlett Packard acquired the company in October 1998, and sold it to PRA Solutions Inc, a subsidiary of Accenture (formerly Andersen Consulting), on 10 November 2000. Delaware law permits the merger of companies with assumption of all rights and liabilities of the merged companies by the resulting entity. Open Skies was

merged into PRA Solutions, and the resulting entity was renamed Navitaire Inc, the claimant in these proceedings.

18. easyJet was one of the first companies to take a licence of the OpenRes system. Mr Ray Webster, the managing director of easyJet, knew Mr David Evans and Mr Neeleman, and approached Mr Neeleman with a view to obtaining a licence, which was granted. easyJet were at the time using a system called AVOPS. AVOPS was a Unix-based system running on PC-type hardware. It was written using a tool called FilePro and was provided by a small company in the United States called Sysops. AVOPS was unreliable and incapable of supporting easyJet's expansion.

The easyJet licence

19. The easyJet licence was signed on behalf of Open Skies inc, and easyJet Airlines on 18 November 1996. It is convenient to set out the relevant provisions in one place.

> 1. **Grant of License.** Subject to the terms and conditions set forth hereafter, Open Skies hereby grants to easyJet a non-exclusive license to use *OpenRes*. (Attached at Exhibit A hereto is a functional description of *OpenRes*). This license shall be perpetual unless terminated by Open Skies in accordance with the provisions of paragraph 12 below.
>
> 2. **License fee** [the licence is fully paid up within thirty days of completion of installation of OpenRes. There is no continuing payment provision]
>
> 3. **Customization of Data Uploading**
>
> a. Attached as Exhibit B hereto is a letter from Farrukh Khan of Open Skies to easyJet dated October 3, 1996. This letter divides customization work that will be required by easyJet into three phases: Phase I, Phase II and Phase III. The [] license fee described in section 2 above shall cover all the Phase I and Phase II customization work described on Exhibit B. Phase III customization, as well as any other customization not described on Exhibit B shall not be covered by the license fee. The charge for such non-covered customization shall be $85 per hour (easyJet may apply the 35 hour per month credit described in section 9 below towards Phase III customization or any other customization desired by easyJet.)
>
> b. The license fee also covers Open Skies' time incurred in uploading data from AVOPS and easyJet's phone system, provided that easyJet supplies such data in the agreed upon format.
>
> 4. **Conditions of License.** This license is granted subject to the following conditions:

> a. Title to and ownership of *OpenRes* shall remain with Open Skies. (easyJet, however, shall own and retain title to all underlying easyJet data generated by *OpenRes*). easyJet acknowledges that *OpenRes* is the property of Open Skies and that easyJet's rights in and to *OpenRes*, or any portion thereof, may not be assigned, licensed, sub-licensed or transferred (whether by operation of law or otherwise) without the prior written consent of Open Skies, which consent shall not be unreasonably withheld. This restriction on transfer shall not apply to (a) a merger in which easyJet is the surviving entity or (b) a change in the ownership or control of easyJet.
>
> b. Because the license being granted to easyJet is non-exclusive, Open Skies retains the right to sell or license *OpenRes* or any portion thereof to any other entity or individual.
>
> c. Any *OpenRes* software that is supplied to easyJet in machine-readable form may be copied, in whole or in part, by easyJet only for back-up or archive purposes. No other form of copying is allowed.
>
> d. easyJet agrees not to disclose or otherwise make available *OpenRes* or any portion thereof, or any related material or information, to any person or entity outside of easyJet without the prior written consent of Open Skies. The granting or withholding of such consent shall be entirely within the discretion of Open Skies; however, such consent shall not be unreasonably withheld by Open Skies. Open Skies agrees that it shall not disclose any commercially sensitive information relating to easyJet without easyJet's prior written consent.
>
> …
>
> 13. The license granted hereunder shall terminate in the event of a default by easyJet that is not cured within thirty days after written notice.

20. Exhibit B to the agreement is concerned with customisation for easyJet's requirements. Phases I, II and III specify the items to be completed before installation, within sixty days of installation and 'as part of a long-term enhancement list' respectively, and each Phase is itself divided into three categories of development. For present purposes it is only necessary to note that Category I of Phase III specifies the development of internet access to the booking engine. This eventually appeared in the

form of TakeFlight, largely developed by a programmer called Justin Wilde. There is no dispute between the parties that TakeFlight was supplied pursuant to the obligation contained in clause 3a and Phase III of Exhibit B.

21. easyJet 'went live' with OpenRes in June 1997. Before that date, all the data on the AVOPS system had to be transferred to the new databases. This was done by Ms Lane Antry's specifying a so-called 'neutral file format'. This format identified the data that was necessary to the operation of OpenRes and accordingly needed to be extracted from AVOPS. As I understand the evidence, the data was then extracted from the AVOPS system using software designed for the purpose into text files of a prescribed format, perhaps 'comma-delimited', in which the data fields are separated by commas. The data was then loaded into the OpenRes database using software that was designed by Ms Antry to expect the data in the prescribed order and separated by the prescribed delimiters.

The OpenRes system

22. Dr Chiu, easyJet's expert, produced a diagram of the functional components of the OpenRes system and the systems associated with it as it was used at easyJet. This is a useful diagram, and was not challenged for accuracy.

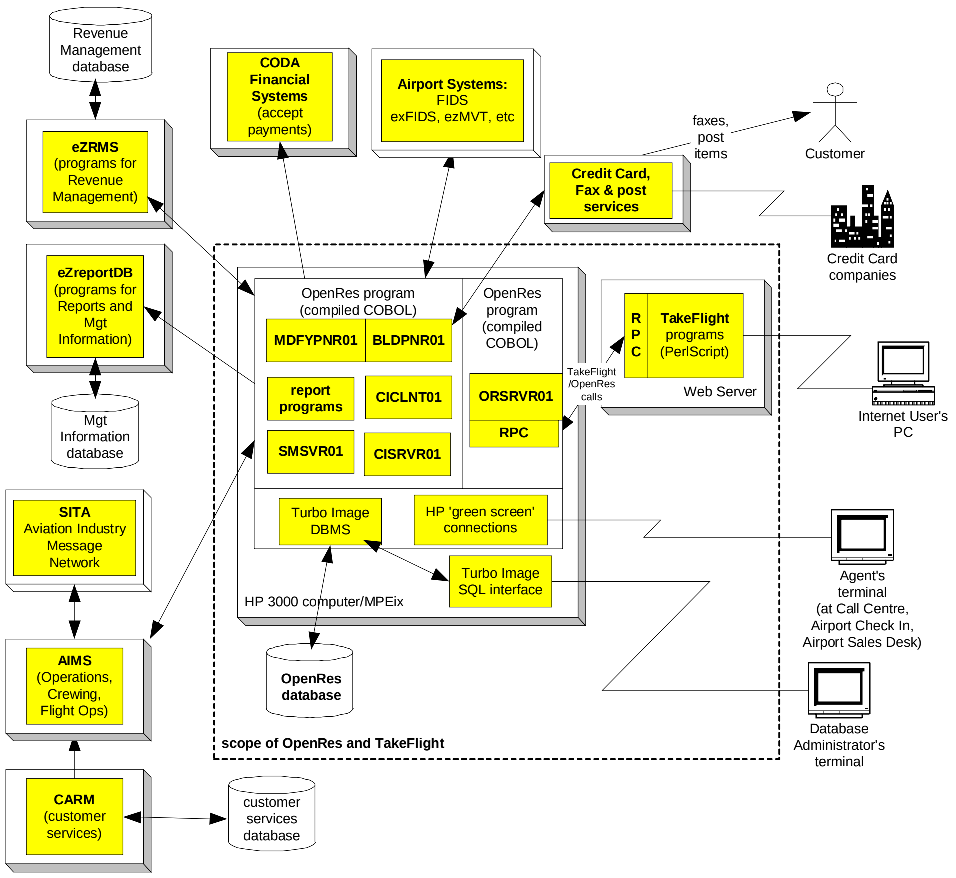

Revenue Management database
eZRMS (programs for Revenue Management)
CODA Financial Systems (accept payments)
Airport Systems: FIDS exFIDS, ezMVT, etc
faxes, post items
Customer
Credit Card, Fax & post services
Credit Card companies
eZreportDB (programs for Reports and Mgt Information)
OpenRes program (compiled COBOL)
MDFYPNR01
BLDPNR01
OpenRes program (compiled COBOL)
report programs
CICLNT01
ORSRVR01
SMSVR01
CISRVR01
RPC
TakeFlight /OpenRes calls
R P C
TakeFlight programs (PerlScript)
Web Server
Internet User's PC
Mgt Information database
SITA Aviation Industry Message Network
Turbo Image DBMS
HP 'green screen' connections
Turbo Image SQL interface
HP 3000 computer/MPEix
Agent's terminal (at Call Centre, Airport Check In, Airport Sales Desk)
AIMS (Operations, Crewing, Flight Ops)
OpenRes database
Database Administrator's terminal
scope of OpenRes and TakeFlight
CARM (customer services)
customer services database

Figure 1: Overall view of OpenRes system

23. The diagram needs some explanation. The OpenRes and TakeFlight systems that form the subject matter of the claim are within the dotted box. Flows of data between the OpenRes system and programs cooperating with it are shown by the black-headed arrows that cross the dotted line. The OpenRes system itself comprises a database and a family of programs that manipulate the data. I am concerned with those parts of the

system that (1) take the commands typed at an Agent's terminal and recognise them, and format the results of those commands to be displayed on the 'green screen' (the 'terminal user interface') (2) the appearance of the graphical user interface at the database adminstrator's terminal (the 'Fares and Scheduling Interface') (3) the appearance of the screens produced at the Internet user's personal computer by the TakeFlight programs (the 'Internet user interface' or 'TakeFlight interface') and (4) the structure of the OpenRes database and the names of certain of the objects that are stored within it.

24. Dr Chiu also produced a diagram of the software modules making up the OpenRes system[1]

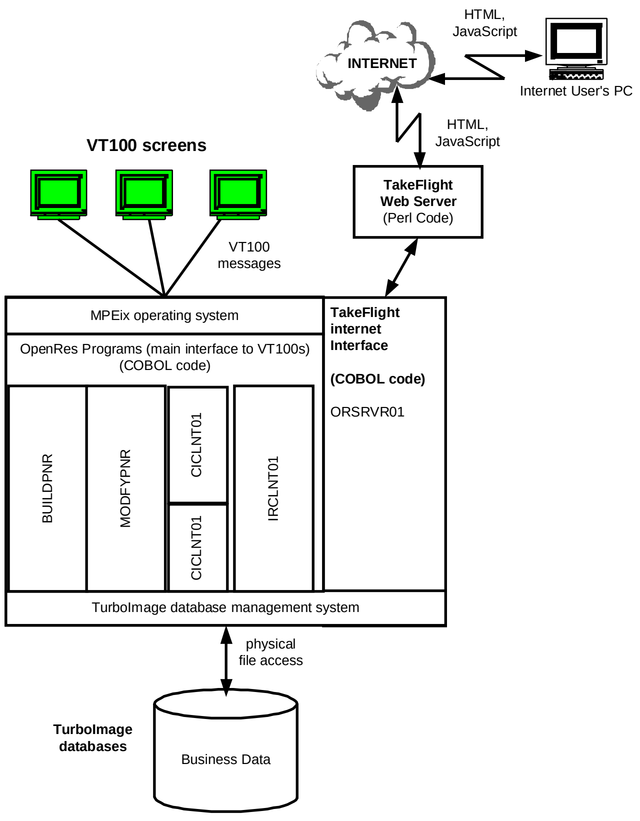


Figure 2: software modules in OpenRes

The user interfaces

25. A great amount of time was spent on the command line interface, more than the difficulty of the subject really justified. During the trial, both the terminal (i.e. non-internet) interface, the Fares and Scheduling interface and the Internet user interface were considered separately. Any user interface of the kind under consideration in this part of the action has two aspects: the commands typed on the keyboard by the user, and the display on the screen. The code defining the user interface is accordingly distributed, in the OpenRes system, between the various software modules that require input from the terminals: BUILDPNR, MODFYPNR, CICLNT01 and

[1] This diagram contains a typographical error. The lower one of the boxes labelled CICLNT01 (Check-In Client 01) should be labelled CISRVR01 (Check-In Server 01) as Dr Hunt pointed out.

IRCLNT01. The first two make up the reservation system; the third the checkin system and the last is concerned with ‘irregular operations’ (‘IROPs’).

26. The pleaded case identifies four classes of relevant copyright works in paragraphs 7.2—7.5 of the Particulars of Claim. In summary, they are:

i) ‘the literary works comprising the title, form and nature of each of the literary codes…’ represented by the user command codes set out in Schedule A and response 14 of the Further information provided by Navitaire on 22 November 2003. I shall refer to these as the ‘individual command sets’, as they have in common that they consist of a common prefix followed by optional suffixes and arguments. Each prefix thus identifies a set of commands. A simple example is the notepad command set. The prefix is NP: if the user types NP↵ at the prompt (I use the ↵ for carriage return) the contents of the ‘notepad’ are displayed on the screen. There are two other notepad commands: NP.↵ which clears the notepad, and NP-↵ which permits the operator to add to it.

ii) The ‘complex commands’. Navitaire divide the ‘literary codes’ into the simple and the complex. They differ chiefly in that the ‘complex’ commands allow varying arguments. Dr Hunt described them this way:

> ‘Complex commands are those where the user enters a mixture of command characters and data and has a number of sub-options or choices. The exact combination of command and data determines the response that the system will give. These complex commands are equivalent to the screen templates or windows that are more commonly used in business applications in that they allow the user to enter values for a number of different data elements at once. These complex commands are set out in Schedule A of the Re-Amended Particulars of Claim.’

The A command (for ‘Availability’) is an example (I take this from Schedule A to the Particulars of Claim):

12	A
12.1	A[departure date][city pair] (optional +[days] .[fare class])
12.2	A[departure date][city pair]/ (optional +[days] .[fare class])
12.3	A[departure date][city pair]/[return date] (optional +[days] .[fare class])
12.4	A[city pair] (optional +[days] .[fare class])
12.5	AA[days]
12.6	AS[days]
12.7	AA[days]/S[days]
12.8	AR
12.9	AR[return date] (optional +[days] .[fare class])
12.10	A[day of week][city pair] (optional +[days] .[fare class])

12.11	A[day of week][city pair]/[return day of week] (optional +[days] .[fare class])
12.12	A[city pair]/[return day of week] (optional +[days] .[fare class])
12.13	A[city pair]/(optional +[days] .[fare class])
12.14	A[city pair]/[return day of week] (optional +[days] .[fare class])
12.15	AA[days]/A[days]
12.16	AS[days]/A[days]
12.17	AS[days]/S[days]

The command character is ‘A’. Dr Hunt’s explanation is somewhat obscure. All it means is that ‘A’ will give you different results depending on what you follow it with. If you follow it with something that is not allowed, you get an error, or should do so. Note that the square brackets denote an entry which is generally obligatory. Thus, if the operator types ‘A’ followed by a date and a city pair, available flights satisfying those criteria will be displayed on the screen. For example, in OpenRes, the command A13JUNLTNAMS↵, where the flight date is 13 June, the originating airport is Luton (LTN) and the destination airport Amsterdam (AMS) should produce a screen displaying the available flights on that day:

```
1 EZ 13JUN Fr 9 103 LTNAMS 1445 /1605 0 L#0035:9 W#0120:9

2 EZ 13JUN Fr 9 204 LTNAMS 1735 /1900 0 L#0035:9 W#0120:9
```

The ‘:9’ is the number of seats available: if more than nine are available, only the digit 9 is displayed.

iii) All the OpenRes user command codes, considered as a ‘compilation’. As I understand this allegation, it is merely an alternative legal basis for supporting the complaint in fact, which is that all the eRes commands are either the same or very similar to the corresponding OpenRes commands.

iv) The layouts of particular screens of the terminal user interface. They are set out in section 5.2 of Dr Hunt’s Report of 2 May 2003. An example (the reservation screen) suffices:

(a) eRes

(3) Reservation Display

```
Agent:1900   Booked:13JUN02 10:39   Modified:13JUN02
EZ Rec Locator E0789UF   Received:GILL  Lang:EN Curr:GBP Dist:A
01-EZ 225 Q 15JUN Sa LTNAMS 0 HK01 1355/1600 Q20.00    GBP    20.00    ->01
Fare:  20.00   Tax:  5.00   Fees:  .00   Tot:  25.00 GBP
          Total-cost        Payments         Balance
GBP :          25.00           25.00           0.00
-----------------------------------------------------
mod:           0.00            0.00            0.00
Names:01
01.HUNT/GILL MRS+INF
Comments:01
G  TERMS AND CONDITIONS EXPLAINED                              0613 1036 1900

Payments:
01)VI_4000300020001000-0603   A  25.00 GBP CC020613 934172 25.00
History on file.
```

(b) OpenRes

(3) Reservation Display

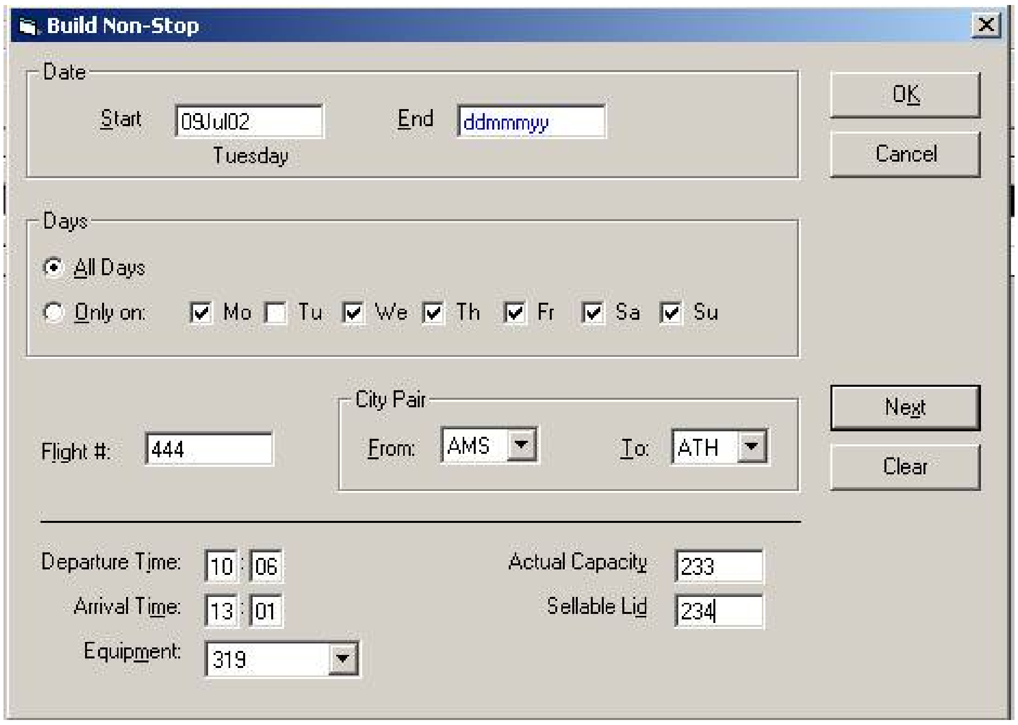
```
Agency:          / Ag:LL     Booked:01MAY03  14:12      Modified:01MAY03
EZ Rec Locator H6C6AG     Received:GILL            Lang:EN Curr:GB  Dist:M
01 EZ  103 L 15JUN Su LTNAMS 0   HK01 1445/1605 L35GB        35.00        35.00
02 EZ  104 L 17JUN Tu AMSLTN 0   HK01 1410/1530 L35GB        35.00        35.00
ADT-GB      70.00  Dep     12.30  Arr       .00  Tot      82.30          82.30
   1 -GB      70.00  Dep     12.30  Arr      0.00  Tot      82.30  GB      82.30

        Total-cost          Payments          Balance
GB :         82.30             82.30             0.00
mod:          0.00              0.00             0.00
Names:01                                     Invoice/IATA#:
 01.HUNT/JOHN MR
Comments:01 + 00
  - ANOTHER ONE                                                 0501 1412 LL
Payments:
01)VI_4938446101286528-0303      $375.00       0.00 CC000723 76490       164.60
History on file.+001
```

v) Certain reporting functions, and the corresponding screens. The term ‘report’ in this context means a display of useful data extracted from the database and formatted in a useful manner, either for immediate reference or for ultimate supply to other systems, such as financial systems.

vi) The Fares and Scheduling interface. This module is provided to enable the database administrator to make long-term alterations to the database, by providing a new fare structure or a new flight schedule. The screens are set out in section 5.3 of Dr Hunt’s report of 2 May 2003. I show her second example: the build non-stop screen, that controls a function that constructs a schedule for a flight, and stores that schedule in the database. The first screen is the eRes screen, and the second the OpenRes screen:

eRes:



Build Non-Stop

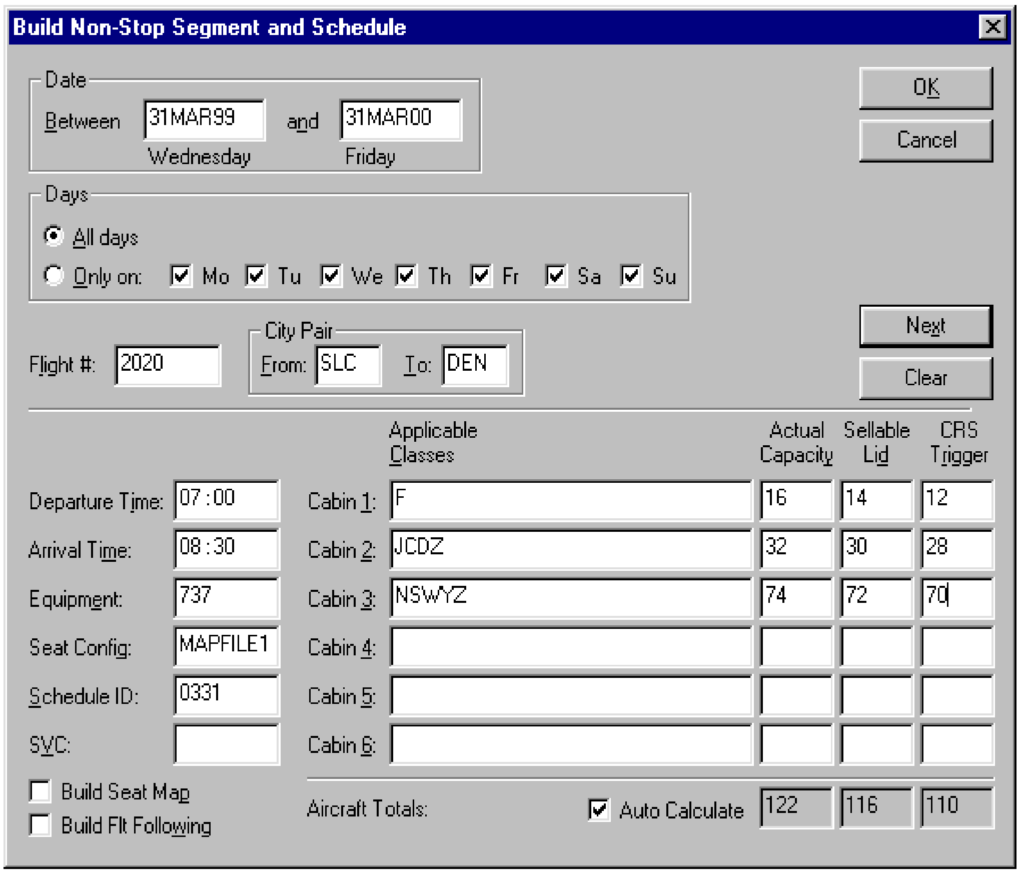
Date
Start 09Jul02 End ddmmmyy
Tuesday

Days
(● All Days)
○ Only on: ☑ Mo ☐ Tu ☑ We ☑ Th ☑ Fr ☑ Sa ☑ Su

Flight #: 444
City Pair
From: AMS To: ATH

Departure Time: 10 : 06
Arrival Time: 13 : 01
Equipment: 319
Actual Capacity 233
Sellable Lid 234

OpenRes:



Build Non-Stop Segment and Schedule

Date
Between 31MAR99 and 31MAR00
Wednesday Friday

Days
(● All days)
○ Only on: ☑ Mo ☑ Tu ☑ We ☑ Th ☑ Fr ☑ Sa ☑ Su

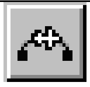
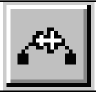
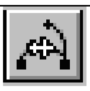
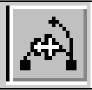
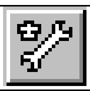
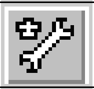
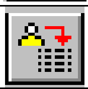
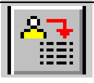
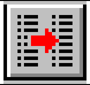
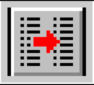
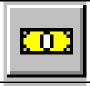
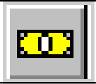
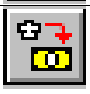
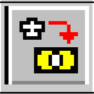
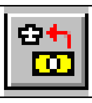
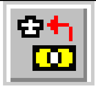
Flight #: 2020
City Pair
From: SLC To: DEN

			Applicable Classes	Actual Capacity	Sellable Lid	CRS Trigger
Departure Time:	07 : 00	Cabin 1:	F	16	14	12
Arrival Time:	08 : 30	Cabin 2:	JCDZ	32	30	28
Equipment:	737	Cabin 3:	NSWYZ	74	72	70
Seat Config:	MAPFILE1	Cabin 4:				
Schedule ID:	0331	Cabin 5:				
SVC:		Cabin 6:				

☐ Build Seat Map
☐ Build Flt Following

Aircraft Totals: ☑ Auto Calculate 122 116 110

The claimant says that the eRes screen contains all the material from the corresponding OpenRes screen that is appropriate for a single cabin aircraft. OpenRes contains facilities that are suitable for airlines using more than one class. easyJet is not such an airline, and the refinements have been omitted. As part of the case on copying in this area, Navitaire also rely on the undoubted fact that the detailed designs on certain of the buttons used on this GUI have been copied, and I set out a table showing them.

OpenRes Schedule Planning	eRes Fares and Scheduling
	
	
	
	
	
	
	
	

Where the commands are listed

27. The pleaded commands are in Schedules A and B to the Particulars of Claim. When Dr Hunt came to prepare her reports, she produced a different table (Appendix 2 to her report of 22 October 20003) which sets the commands out in a different order. It is Dr Hunt's Appendix 2 which is cross-referenced to a large collection of code fragments from OpenRes and eRes that are intended to show how the codes are themselves expressed in software. The OpenRes fragments were made available with the 22 October report: the eRes fragments followed in January with her second supplementary report.

How the commands work

28. This is not the place comprehensively to describe the manner in which digital computers work. It is still necessary to examine certain aspects in a little detail. The call centre operative, or airline booking clerk, is provided with a screen and a keyboard. We are not here concerned with a screen capable of displaying complex graphics but an 80 by 24 character standard green screen. Most such terminals respond to control codes devised by Digital Equipment Corporation for its VT100 terminal, and are generically referred to as VT100 terminals. When the terminal is switched on a program running on the computer to which the terminal is connected prompts the operator for a password. Then he or she is given an introductory screen, listing the various functions that may need to be undertaken. Once the operator has selected the type of operation, an appropriate prompt will be presented, inviting the operator to type a command. He or she wants to know the availability of seats on

flights from Luton to Amsterdam on 13 June, and types ‘A13JUNLTNAMS↵’. What then happens?

29. The answer is that it depends upon what the author of the software wrote. There are a myriad ways of constructing a program or part of a program (a ‘parser routine’) that takes a line of input, breaks it into its component parts, interprets the command as requiring certain data relating to all flights with free seats between Luton and Amsterdam on 13 June, calls appropriate routines to produce that data from persistent stored data (the database), formats it and causes it to be displayed. What the software *does* is recognise the whole command: Navitaire originally proposed that it was not necessary to examine the software at all. All that was necessary was to see what the command did. In the end, there was evidence as to what the software did, but it was produced mainly in cross-examination, although Dr Hunt had, by the end of the trial, identified the parts of the code that contained the parsing routines for all the commands alleged to have been copied in both systems. Thus, when talking of parsing a city pair (LTNAMS in my example) Dr Hunt said this:

> ‘Q. Nowhere in the code is the fact that that part of the argument at the A command is going to be a city pair is recorded. It is just what the code does. Do you understand the distinction which I am drawing? A. I think I do. There is nowhere in the code where it says a city pair in those terms. What it says is when A, treat this as a city pair.
>
> Q. Treat what is coming as a city pair, but what it does is it inserts an appropriate code to read the next six characters, assemble them on the assumption they are a city pair. A. Yes.
>
> Q. And then do whatever it needs to validate them as a city pair. A. Yes.’

30. This answer was given to a question about the eRes system, but the position is no different for OpenRes. Mr David Evans was taken through the parsing of an availability command in words, without showing him the code, in re-examination. He described the process:

> Q. Let us look at just identification of the leading A. How was it done? A. It would look for an A, it would fall to a case logic, so it is ----
>
> Q. You would have a branch statement, or something like that, a series of cases. A. A series of cases and the first case would be A.
>
> Q. So it would be case A, case B, case C, case D, and so on. A. That is correct.
>
> Q. Each of them would have a branch, or whatever, underneath it. At that point, it would branch out to the appropriate sub-[process]. Let us look at the next in 12.7. You have got a date. The date was entered by the operator in a particular format. Is that right? That is to say, year, month, date. Have I got it the

wrong way round? A. It was actually just the day and the month.

Q. That was the difference, in fact. It was just the day and the month. OK. We are now in the [process] appropriate for A commands. How did you parse the string corresponding to a date? Do you remember? A. I would find the bits that are like the 30 APR. I would pass them to a function and say, "Now, tell, me first of all, what year they are referring to?", and then I would make an 8 digit date out of it.

Q. How did you make the 8 digit date? A. Using a routine to figure out what the month number was and with that I would go through some logic to say OK, is this month before the month we are in today? If it is, I am assuming that is next year's date and then I would string it all together and I return, "here is a date".

Q. So if we are in November and we have got somebody saying October, we can reasonably assume it is October next year. A. Yes.

Q. So we have got the year. A. Something like that, yes.

Q. October. It is typed in by the operator as OCT. How was the how did the code recognise an OCT? As it happens, it is the only month that begins with O, but that does not matter. How was it actually done? A. All the months would have been in a string, so JAN ----

Q. Each of them consecutively? A. Consecutively. You do a positional to it and then divide by or something and get the number.

Q. You jump down. You move down frame by frame, as it were? A. Yes. There is a function find the position of where OCT is in the string and derive the number from that.

Q. Somewhere there was a fixed things with things to compare? A. Yes.

Q. We now have the date. We now have a city [pair]. The city pair is typed in again as a 6 character, IATA standard names for the two airports. A. That is correct.

Q. How did you parse the city pair, do you remember? A. I think I would grab it as a city pair.

Q. You would just grab it as a city pair? A. Yes.

Q. Without further processing? A. Without further processing.

> Q. So, in fact, all you do is just drop that one straight through because you know that that is what you need and it is readable as a city pair? A. Yes.
>
> Q. Then you have a return date. That is parsed in the same way As we have already discussed dates being parsed, and it is separated by a slash. How do you recognise the slash? A. With a positional function.'

31. I am not sure that I was ever taken through the actual code that David Evans is describing by one of witnesses. It is to be found in the file labelled 26.txt in the code fragments produced by Dr Hunt to accompany Appendix 2 to her October report.[2] I have copied the relevant passage from the source code for OpenRes 5.58 and I have set out the section corresponding to Mr David Evans's evidence in Annexe 1 below.

32. Generally, a digital computer required to parse a line of input text will analyse the line according to rules that are implicit in the software that controls the parsing operation. The computers with which this action is concerned parse their input sequentially. It is the task of the programmer to provide code which controls the machine to analyse the line correctly and act as required in response to what the line contains. We have in this case a large number of commands, generally identified by their first letter or other symbol. In OpenRes, the Reservations module contains a command for either sorting lines on the screen ('/A') or for inserting a line after a particular segment displayed on the screen ('/*n*M*m*'), for displaying certain information (first character '.'), for providing help ('?'), for displaying available flights ('A'), for entering comments in respect of a particular reservation ('C'), for displaying a calendar ('CAL'), for doing calculations ('CC'), for payment ('F'), for getting/selling seats ('G') and so on. So, what does the programmer do? In OpenRes, the code starts by analysing the first character of the string typed at the VT100 terminal. In Annexe 2 below, I set out a code fragment that does this and provide my understanding of it.

33. What cannot be found in the OpenRes code is a record of the commands in anything like the form in which they appear in Dr Hunt's Schedule 2 and the annexes to the Particulars of Claim. The name of the command will appear if it is a single letter. If the name of the command has two or more letters, it may or may not appear recognisably. A further code extract may make this clear. This is part of the 'a-commands' section of code which is entered when the first letter is 'A':

[redacted]

At line 263400, the whole line is checked to see if the first 5 characters are 'ARNK/'. This is the whole of a command (amended Schedule 2 number 13.) The programmer

[2] The evidential position on these code fragments is confusing. When Dr Hunt's October report was prepared, it seems to have been provided with a CD-ROM containing all the fragments. In Appendix 2, each command code was numbered. Each corresponding source code fragment had a filename including the corresponding number. Thus the availability command was numbered 26, and the file containing the source code is 26.txt. Unfortunately, the Appendix was amended and the numbers were changed. When the eRes code fragments implementing the commands were prepared, they were numbered according to the amended numbering. So the Availability command in eRes is in file 9.txt. The eRes fragments are all to be found in the trial bundles at F3A[14].

also checks for the whole of the address prompt command 'A-' (amended Schedule 2 number 10)[3]. But he doesn't bother to check for the whole of the additional information command 'AI' (amended Schedule 2 number 11)[4], perhaps because he has remembered that at this point in the code the first letter must be A. So he just checks the second, to see if it is I. The consequence is that 'AI' is never recorded in the code as such. It is a consequence of running the code that if a command line beginning AI is processed, the 'display-additional-info' code is performed.

34. The same point may be seen in the code extract in Annexe 2 in the processing of the 'F' commands. The 'F--' (amended Schedule 2 number 48)[5] is never expressly recorded as such in the code. At lines 198200-198300, two checks for a dash are performed: the first on position 2 and the second on position 3. This is done so that a negative answer to the second check can be followed by a check for an 'H' in the third position, indicating the 'F-H' command (number 49)[6].

35. As the purpose of the software is to analyse the commands to see what has been typed and execute the appropriate action, it is perfectly irrelevant to *function* whether the codes are all set out expressly somewhere or not. (In eRes they are all set out expressly, because the parsing technique is quite different, but I shall return to that.) It does, however, mean that the command code can only be identified as such from the software itself by working out what strings of letters and symbols it recognises as command codes: in other words, by seeing how it works in use. I have read through the various input routines in BUILDPNR and CICLNTT01 to see to what extent the codes are present expressly. In the majority of cases they are, but in some they are not. Whether they are or not is irrelevant to whether the code works to recognise the commands: it is a different way of analysing the input to see if it contains a particular command.

36. The 'syntax' or permitted arguments of the various commands and sub-commands is not explicit in the code. This is invariably the case. It follows that if the syntax of the command is not recorded in some manual or help screen, the only way of determining the syntax of a given command is to analyse how the code operates in use. As easyJet submits, the syntax of the various commands is not recorded in the source code; rather the source code recognises and acts on permitted arguments to a given command.

37. An analysis of Mr David Evans's evidence, with that of the experts and consideration of the code itself satisfies me that OpenRes does not contain any text corresponding to the commands as pleaded. What it does contain is code which, when executed by the computer, will accept commands with the particular arguments specified and produce the specified results. As part of the code, the individual letters and other symbols ('.', '/' and so on) will appear expressly somewhere in the code, and the whole name of the command will often do so, in the section of code in which the components of the command entered at the command prompt are analysed. With this introduction, I can turn to the story of the commands.

[3] Not alleged to be copied.
[4] Not alleged to be copied.
[5] Corresponding command present in eRes, which treats it as part of the two-character command 'F-'.
[6] Not alleged to be copied.

Development of the commands

38. It has been surprisingly difficult to list all the commands in the OpenRes system. Indeed, Schedule A to the Particulars of Claim, which lists the 'complex' commands, has been amended four times, and the list of all commands contained in Schedule B has been amended five times. It is not clear that even now the lists are entirely accurate, but this seems to me to be a matter of slight importance. The codes were mainly designed by Mary Beesley, who gave evidence. She was not responsible for writing the software to implement these commands. David Evans coded the commands for the Reservations module, Greg McDaniel those for the Check-In module and Mike Padgen those for the IROP module.

39. The word 'brainstorming' is used some forty times in the witness statements in support of Navitaire's case, and is always used either in respect of the design of a command or of a display. The process is described by Lane Antry, a systems designer:

> 'During the course of the last 9 years of involvement with OpenSkies I have performed both of the previously mentioned roles. I have spent extensive time on-site with our customers, during which I have worked directly with them to identify their particular business needs. Then I would turn this 'needs assessment' into a form of business processing logic (generally in collaboration with other team members) in order to formulate detailed requirements for the programming staff. This step in the process is where we would brainstorm and flesh out the details, including the logic, of each step or function required in order to meet the customer's specific business needs. The output of this process would typically result in the creation of a particular command syntax, input requirements, output formats, (either report or screen layouts), the need for a particular command sequence or not, and often specific recommendations regarding database storage and retrieval locations. The result of this process would generally be communicated orally to a particular programmer who may have been part of the brainstorming team. They would then write the computer program. Sometimes however, the results or portions of the results would be written into a more formal specification document. Often times, I would then actually perform the programming duties myself, due to my previous years of experience as a programmer.'

40. There is in my judgment no doubt that the OpenRes user interface was influenced by other reservation system command line interfaces. Ms Beesley, who was intimately involved with the development of OpenRes, had a good knowledge of other systems. David Evans was himself well acquainted with SABRE, which remains a widely used system. Command line interfaces are common. OpenRes's origins are in MARS, which had an interface that was in part derived from SABRE. Indeed, Mark Sapitsky, who worked with David Evans in its design, wanted the whole of the command set to be the same as SABRE, but Ms Beesley thought many of the latter's commands to be excessively complex and in those cases, a simplified or altered command was included in MARS. Sometimes, she says, both the new command and the SABRE command were incorporated. There is a hangover from this in OpenRes (and, given

the copying of the user interface, in eRes also), which is that S is a synonym for A, although the reason is not explained: it too provides an availability command.

41. MARS was not, of course, a ticketless system. Most of the commands are entirely independent of the nature of the system, but some are not. Availability is not affected by whether tickets are to be printed or not, and there is a general similarity with other systems, such as Galileo and Amadeus, which is obvious. The commands do differ between systems (apart from OpenRes and eRes); but the essence of an availability command, for example, lies in the operator's ability to specify a date, a departure point and a destination. They all have this, and the refinements, such as ranges of dates, return flights, class, maximum fare the customer is willing to pay, numbers of passengers and so on come later. Ms Beesley's evidence was not entirely consistent, but there is no doubt that what the Open Skies programmers produced was another command line interface, with the same general capabilities as other command line interfaces had, over a system expressly orientated to ticketless travel. The evidence is quite clear that it is in this business model and in its implementation, that the real innovation of OpenRes lies.

42. I have no doubt that it is proper to refer to a 'command code design process' and that the specification of a command would, as Ms Beesley describes, start with the function to be performed, as it is normal to start with the result to be achieved. Navitaire's evidence as to who was responsible for the design of the various commands is uncertain. I reject any suggestion that the entire command set was systematically designed. Although the commands obviously form functional groups, there is little evidence of systematic design over and above the necessity of avoiding duplication and ambiguity when a new command is added. It is very difficult to discern any overall structure to the codes, other than that they implement a particular way of doing business, or business model.

43. I am asked by Navitaire to make a number of findings in respect of the user commands. I set out the proposed findings with my comments.

44. I should observe that all these proposed findings are loaded with terms (author, skill and labour, compilation and so on) that might be thought to assume that copyright subsisted in the commands set out either in Schedule A or in the collection of all the commands set out in Dr Hunt's Annex 2, and I have attempted to avoid using these terms because one of the principal issues in the case is whether copyright does subsist in the matters set out in these documents.

'1. Each of the complex commands constitutes a family or collection of sub-commands based upon one main command code. All of the individual commands within the family are inter-related or interlinked in their expression, syntax and functional meaning.'

45. This is true, but repetitive. It is also a statement of how the commands appear to the user, rather than how they are parsed, which is not so well defined. The phrase 'inter-related or interlinked in their expression, syntax and functional meaning' is just another way of saying they are a family of commands, and adds nothing.

'2. Each of the commands was encoded in the same section of the source code.'

46. This is not generally correct. I have already explained (paragraph 33) that an availability command such as A13JUNLTNAMS↵ is parsed in the BUILDPNR module for the purpose of identifying it as such. When the display-schedule routine is called (line 264500), control is passed to the AVAILSCR module, which contains the code that recognises the 'syntax', i.e. the arguments, of the command in the manner described in Annexe 2. Of course, it does depend on what 'the same section of the source code' means. If all it means is that once the command code is recognised, the arguments of that command are processed together then it is quite correct, but it is difficult to see why it matters.

'3. Each complex command required skill and judgment in the choice of letters, symbols and syntax to represent the main command and the inter-related family members. Thus when devising sub-commands or making changes and additions each sub-command had to be consistent with the other family members, so that the compilation remained easy to use and memorable to the user as a family group.'

47. I have no doubt that more than negligible skill and judgment went into specifying the codes in general. Some of the complex codes ('A' for availability, 'G' for grab/sell, 'N-' to name a passenger, '.' for print, 'CC' for calculator[7], and so on) do not, on their own, seem to me to require much skill or labour. Ms Antry accepted that QEP[8] is common in the business, as are E, ER, I and IR. As Dr Chiu demonstrated, Galileo and Amadeus, the mainframe systems, use 'A' for availability. Galileo, Amadeus and Sabre all use N ('Needs') rather than G (grab) for the grab/sell command. Passenger names are provided by the 'N.*mm*' command in Galileo (*mm* is the number of passengers), the 'NM*mm*' command in Amadeus, '-*mm*' in Sabre, 'N-' in OpenRes and 'N-*mm*' or 'NT*mm*' in eRes. The '.' command made its appearance in EARS as an alternative for the '*' command, which appears to be generally used by others, because the keyboards used by the Belgian airline for which EARS was written did not provide a convenient asterisk key. All these commands differ, to varying extents, in their detailed syntax. It goes without saying that the syntax should be consistent across sub-commands, but no doubt it requires some skill to achieve that but the fact that the development of the commands was in some cases *ad hoc* can be seen by considering the names of the author(s) and dates claimed for the individual commands and sub-commands claimed in Dr Hunt's Annex 2, where the clearest example is the '.' command. The evidence as to ease of use and memorability was not extensive. I suspect that ease of use comes in any such case with familiarity: even the most intractable command line interface becomes easy to use as it becomes familiar to the user. Some of the commands (DATE, HELP, ?, CAL, EXIT) are entirely standard across many platforms over many years and are not original to the claimants at all.

'4. All of the particular complex commands listed in the Re-Re-Re-Amended Schedule A were created in the manner set out above [sc by Ms Beesley, and

[7] I do not think that the CC command can be described as a complex command. It is just a desk calculator: CCn+m returns the result of adding n to m, and so on. But this does not affect the general argument.
[8] QEP is classified both as complex and as simple in Schedule 2, depending on whether it is in the QEP form or the QEP/ form.

programmed by David Evans (Reservations module), Greg McDaniel (Check-in) and Mike Padgen (IROP)] and by the authors listed in the Schedule.'

48. As I have indicated, the history of the originators of the commands is not entirely clear. I find myself, however, able to say that if such-and-such a command was not originated by the person(s) specified in Schedule A as the author(s) of that command, it was originated by some other person, at about the same time, who was one of the other persons specified.

'(5) Each of the complex commands was original in that as a family of main and sub-commands it was not copied from any pre-existing command, nor as a matter of fact was it the same to any pre-existing complex command.'

49. Again this is generally correct of the commands listed in Schedule A, but the remarks that I have made above apply equally here. I do not accept it as true in respect of the standard commands I have identified, and, of course, many resemble the commands in other systems more-or-less closely. It would be idle to suggest that this command set owed nothing to the pre-existing command sets of which the authors were aware.

'(6) The totality of the command codes in IROPS was objectively novel and unique as well as original'.

50. I have not so far discussed the IROPs code as it seems to me to raise no separate issue of principle. 'IROPs' stands for irregular operations, and covers a series of commands that are designed to carry out bulk alterations, such as assigning a group of passengers already booked to a different flight. IROPs is an airport system. As far as the evidence goes, OpenRes is the only system with bulk commands of this kind. They were designed to operate closely with the commands concerned with check-in in the Check-In module, but on the evidence this suggested finding is justified.

'(7) The collection of all the user commands as set out in the Re-Re-Re-Re-Amended Schedule B was original, was not copied from any pre-existing group of commands and involved creative skill and labour by the authors set out in the schedule.'

51. The emphasis of this proposed finding is on the *collection* of commands. This is meaningful only if the collection constitutes a whole that is more than the sum of its parts, and this would be the case if the assembly of the parts into the whole were inspired by some governing criterion. It is tempting to use the phrase 'planned whole' but that is probably putting the criterion too high. Again, concealed in this proposed finding is a submission that the collection of commands amounts to a compilation in the copyright sense. The whole of the evidence was quite contrary to any such suggestion. The members of the collection were added *ad hoc* on the basis of the knowledge of existing systems possessed by its various designers, and their need to construct new commands appropriate to a ticketless reservation system suitable for smaller low-cost airlines. No command set was ever designed. I do not think it was born: it just growed. It does not, as a collection, represent a single exercise of taste and judgement.

52. Navitaire emphasise that it was not suggested (and cannot be) that the whole instruction set was copied from any other system. This is true, and it is true also that few commands were copied in their entirely. I have given a summary of the main

points in common with pre-existing systems above. I readily accept this, but for the reasons I have given I cannot accept that there is here a 'command set'.

'(8) Skill and labour had been expended in creating each of the complex commands and in the compilation of all the commands in OpenRes.'

53. I have accepted that this is true in general. Any design of a command may involve judgement whether to make it a sub-command of an existing command or a fresh command.

54. I shall consider the consequences of these conclusions after I have considered the eRes system commands and the screen layouts in both systems. I consider that it is necessary to consider all the aspects of the user interface together, and although it complicates the structure of the judgment I cannot at present see any alternative.

THE OpenRes SYSTEM: THE DISPLAY ASPECTS OF THE USER INTERFACE

55. In response to any command information will be displayed on the screen. The format in which information is displayed is also in issue (I have given an example in paragraph 26.iv) above). I think, with Ms Antry, that the screens and the commands really have to be considered together, as the system is structured so that each successive command is entered in response to what is displayed in response to the previous command. This is well illustrated in the so-called walk-throughs that were prepared by Dr Hunt.[9]

56. The VT100 screen display formats (i.e. the Reservations, Check-in and IROPs) have two aspects: fixed data and the position of variable data. The six Schedule Planning screens are of the familiar Microsoft Windows GUI type, and raise different considerations.

57. The evidence in relation to the devising of the screens is divided between Ms Beesley, David Evans, Greg McDaniel, Lane Antry, and Mike Padgen, the last-named being primarily responsible for the Schedule Planning GUI. Others are said also to have contributed. Twenty-seven screens, all identified in Amended Schedule C to the Particulars of Claim are relied on. Some but not all of those screens are set out by Dr Hunt in section 5 of her 'Similarities' report, the remainder being found at various places in the very long Section 4 of that report. Section 4 is said to demonstrate the 'Business Logic' of OpenRes and eRes, and I shall deal with it below. Unfortunately the degree of similarity between the specified corresponding OpenRes and eRes screens varies widely and they cannot be considered globally. Dr Hunt ascribes percentage scores for similarities in her September report at section 9. Cross-examination revealed that the percentage scores merely reflected a qualitative assessment of similarity and that the scores were accordingly unhelpful.

58. There are a number of general observations that can be made. The content of each of the screens to be displayed in response to a command was, on the evidence, discussed and from time to time sketched out on paper. Ms Beesley said:

[9] These were provided in the form of PowerPoint slide shows of screen shots of the terminal interface of both systems. The files I have used are eRes.ppt and OpenRes.ppt, supplied to me as agreed documents.

286. As explained above, I also assisted in the production of the screen layouts and in the layout of most of the reports listed in Amended Schedule C and Schedule E respectively to the draft Re-Amended Particulars of Claim. I sometimes drew sketches which I talked through with the programmer, and the programmer then prepared a mock-up of the screen layout. The mock-up was a screen with no interaction with the database. The programmer would then give me a mock-up of the screen, and I would comment on it. If I had further suggestions, the programmer would go away and amend the code until we were happy with the final result.

287. By way of a specific example, I refer to the .incomplete reservation" screen at page 5.4 of Dr. Hunt's Report of 2 May 2003. I designed this screen with Dave Evans when EARS was created in October 1994 when we were working on the process for booking a flight, and I subsequently worked with Greg McDaniel in early 1996 to update the screen in OpenRes to take into account the order in which the information would be collected during a call. As explained in section F above, we first discussed on a general level the facility that we wanted to introduce, and I then visualised how this would work in practice on a step by step basis. During this brainstorming process, we discussed what should appear on screen, and we often sketched out our ideas.

288. When we discussed the reservations process, I knew from my own experience that there is certain information that an agent needs to input into the system, such as the passenger's name and address. When we worked out the different types of information that needed to be input, I realised that there would be a number of different commands for these, and so thought that it would be a good idea to display on screen the commands used for bookings as an *aide-mémoire* for the agents. For example we first needed to input the language for the booking and the itinerary, then the name, phone, address and so on. I thought it would be helpful to list these commands in basic booking order, so when telephone agents were taught how to make a reservation, the commands would be in the relevant order. This screen could be customised for each customer. For example, if an airline wanted to have a 'comment' section at the beginning (such as a comment to advise the customer what time to check in) then the comments section could be inserted higher up the list.

289. We probably discussed whether the system should prompt the agent to input the different information. We decided against this, as I knew from my own experience that customers can impart this information at different stages, and prompts do not offer this level of flexibility. Therefore I asked Dave to set up the system so that the agents can enter these commands in any order. Displaying all the information that was required to

> complete a booking was a new concept. I am not aware of any other system which does this.

59. The evidence taken as a whole is not satisfactory. It is important to notice, however, that *some* skill and effort went into the question of what the screen should look like, but that the manner in which the result was achieved was not the responsibility of Ms Beesley (or the others providing input as to requirements) but the responsibility of the programmer, who had to write the code to display the results in the required format.

60. I was only shown the code that gave rise to the fixed part of the screen display for any command in Dr Chiu's evidence in his third witness statement. This is a lengthy passage, and I cannot set it out here. There was no effective challenge to what he said. In the example that he gave which is the 'incomplete reservation screen' that is the first item in section 5 of Dr Hunt's report. His conclusions (paragraphs 8.24-26) were not effectively challenged. The cross-examination concentrated on the fact that the spaces and letters of the fixed data (eg. `••Currency code:, ••Language code`) are present in the code. The spaces (I have denoted them with a '•') will print on the line on the screen as they appear on the page in the code. Which lines appear depends upon where the program is in execution and on the data input—see Dr Chiu's third report, paragraph 8.24.

THE eRes SYSTEM: OVERALL STRUCTURE

61. Dr Chiu again provided a useful diagram showing the overall structure of the eRes system.

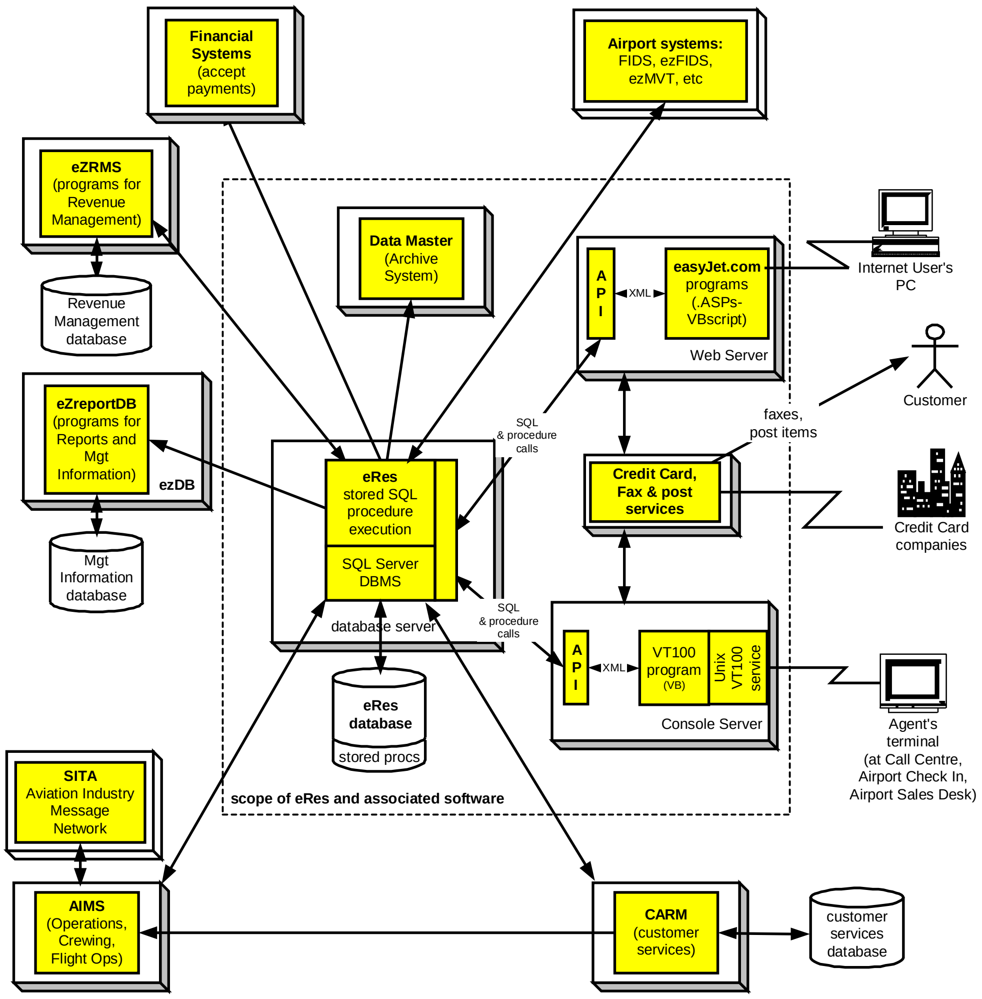

Financial Systems (accept payments)
Airport systems: FIDS, ezFIDS, ezMVT, etc
eZRMS (programs for Revenue Management)
Revenue Management database
Data Master (Archive System)
A P I
XML
easyJet.com programs (.ASPs-VBscript)
Web Server
Internet User's PC
Customer
eZreportDB (programs for Reports and Mgt Information)
ezDB
Mgt Information database
SQL & procedure calls
faxes, post items
Credit Card, Fax & post services
Credit Card companies
eRes stored SQL procedure execution
SQL Server DBMS
database server
SQL & procedure calls
A P I
XML
VT100 program (VB)
Unix VT100 service
Console Server
Agent's terminal (at Call Centre, Airport Check In, Airport Sales Desk)
eRes database
stored procs
scope of eRes and associated software
SITA Aviation Industry Message Network
AIMS (Operations, Crewing, Flight Ops)
CARM (customer services)
customer services database

Figure 3: Overall view of the eRes system

62. Again, the case is only concerned with what is inside the dotted box. Outside the box are the existing systems at easyJet that were accommodated by the new system. The fact that these systems were required to be substantially unaltered with the introduction of eRes imposed certain constraints on the new system, but, as the double-headed arrows show, these constraints (so far as they exist) relate to the database, with which I deal below. They do not constrain the user interfaces in any way.

63. Dr Chiu also produced a diagram of the software modules in eRes (compare Figure 2 above).

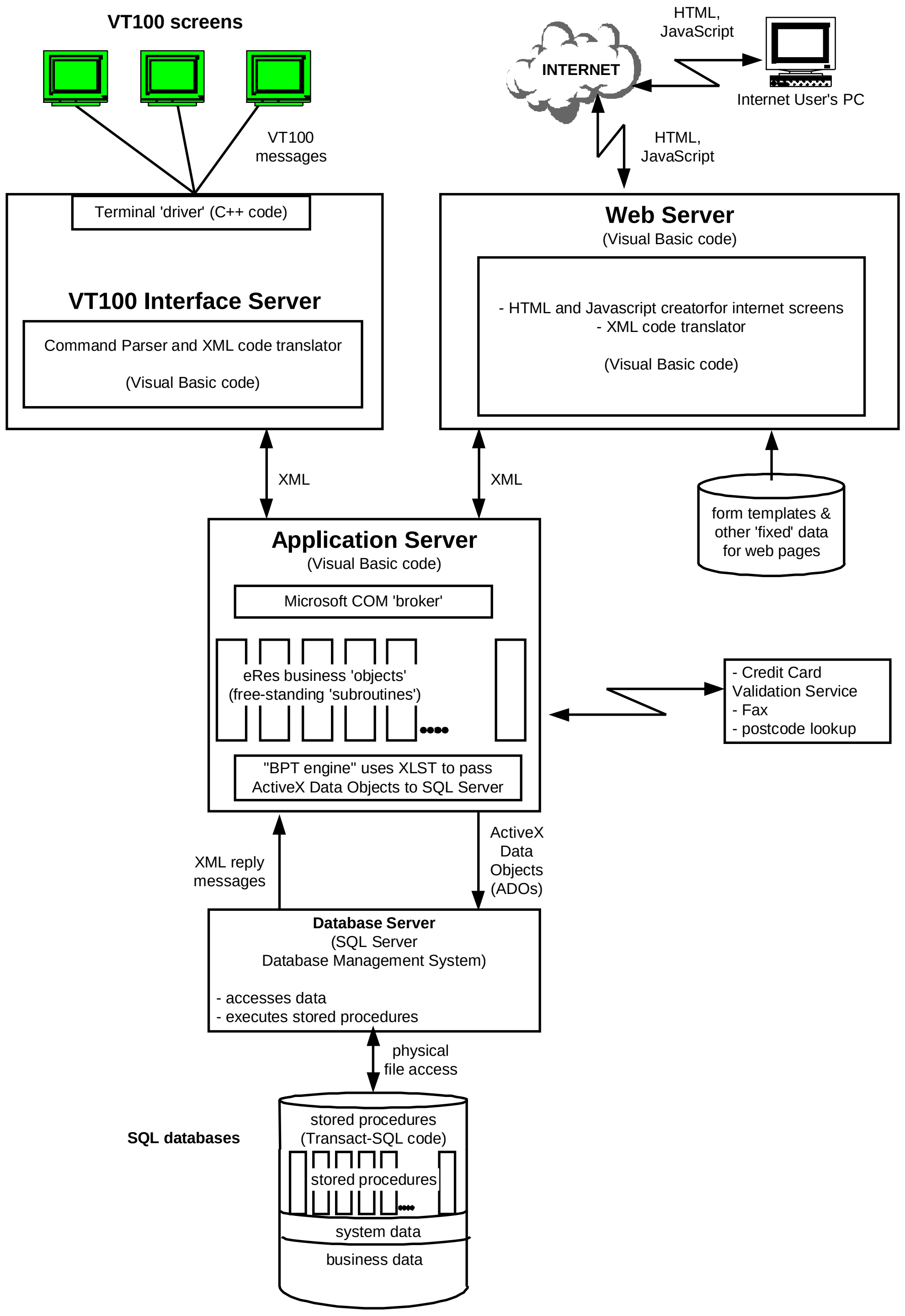

VT100 screens
VT100 messages
Terminal 'driver' (C++ code)
VT100 Interface Server
Command Parser and XML code translator
(Visual Basic code)
INTERNET
HTML, JavaScript
Internet User's PC
HTML, JavaScript
Web Server
(Visual Basic code)
- HTML and Javascript creatorfor internet screens
- XML code translator
(Visual Basic code)
XML
XML
form templates & other 'fixed' data for web pages
Application Server
(Visual Basic code)
Microsoft COM 'broker'
eRes business 'objects' (free-standing 'subroutines')
"BPT engine" uses XLST to pass ActiveX Data Objects to SQL Server
- Credit Card Validation Service
- Fax
- postcode lookup
XML reply messages
ActiveX Data Objects (ADOs)
Database Server
(SQL Server Database Management System)
- accesses data
- executes stored procedures
physical file access
SQL databases
stored procedures (Transact-SQL code)
stored procedures
system data
business data

Figure 4: Software modules in eRes

64. All the command parsing for the VT100 terminals is carried out in the VT100 Interface server. The system makes extensive use of so-called ‘client-server’ architecture, and the application server communicates with its clients (the VT100 Interface server and the Web server) using a uniform Application Programming Interface (API). What happens is that when the VT100 interface server has interpreted a command typed by the user (such as A0113JUNLTNAMS↵) it creates a message in extensible mark-up language (XML) which is then transmitted to the appropriate ‘workflow’ in the application server. The application server does what is required by the message, and returns the result, also in XML. The received message is interpreted by the VT100 Interface server and the results displayed.

65. There is no similarity relied on by Navitaire between the actual code for the user interface parts of the software structures shown in Figure 2 and Figure 4. What is relied on is the fact that the eRes software implements an interface that is very similar, in the aspects relevant to the way that easyJet does business, to that of OpenRes.

THE eRes SYSTEM: COMMANDS AND USER INTERFACE

66. Some 44% or thereabouts of the OpenRes command set has been reproduced, with some variations, in eRes. The principal differences lie in two areas. The first is that the 'core' availability command differs between the two systems in that eRes requires the number of passengers to be input after the 'A', while OpenRes has no such requirement. Where there is more than one passenger (not covered by Dr Hunt's walkthroughs) that affects the way the enquiry and the ensuing transaction are effected. The second is that optional arguments to commands, and whole commands, for which easyJet has no use are omitted. Dr Chiu produced 'syntax diagrams' that I found much easier to understand than lists of possible options for the commands in the two systems:

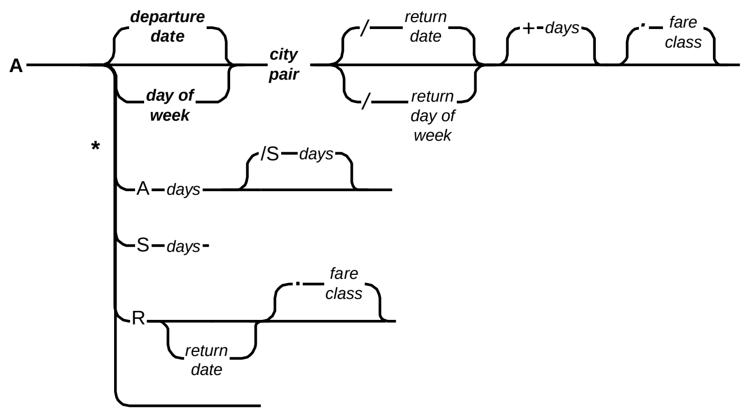


Figure 5: OpenRes availability syntax diagram

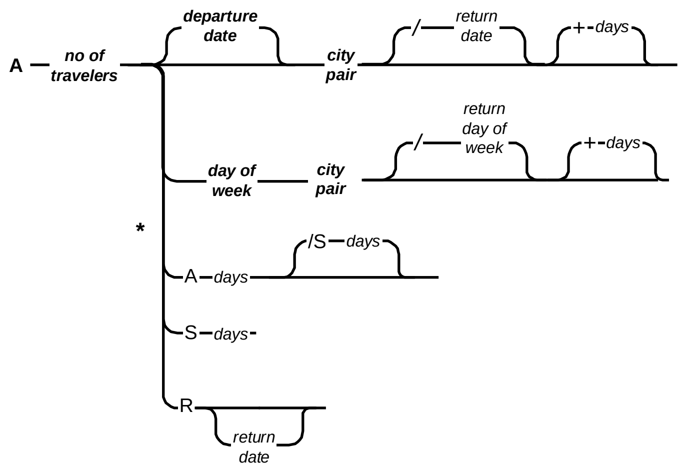


Figure 6: eRes availability syntax diagram

67. Comparing the two diagrams shows, first, that there is a requirement that the number of travellers be specified in eRes but not OpenRes, and second, that the eRes command has been somewhat simplified by omitting the optional '.[fare class]', because easyJet do not have different fare classes. As I understand the evidence, that is entirely typical of the eRes interface. The commands and options omitted are those for which easyJet has no commercial use.

68. The commands typed at the terminal by the user of eRes are parsed by the VT100 Interface server. The relevant pieces of code are contained in Appendix 14 to Dr Hunt's report of 21 January 2004 (F3A[14]). Annexe 3 below explains, again by reference to the 'A' availability command how, in outline, a command is parsed.

69. The names of all the commands appear explicitly in the eRes code in the EzVT100CmdParser.cls source code module. The syntax of the commands is never expressly described: it is implicit in the techniques used by the programmer to analyse the command entered at the keyboard. The result of the parsing is always a message in XML that is sent to the Application Server. I reject the suggestion advanced by Navitaire[10] that the 'portions of the source code of eRes in which the respective commands exist' are to be found set out in Appendix 14 of Dr Hunt's second supplemental report. The commands do not 'exist' as identifiable objects in the code. Appendix 14 contains a great deal of code (some portions of which are repeated many times) that recognises the commands, but that is all.

THE eRes SYSTEM: THE DISPLAY ASPECTS OF THE USER INTERFACE

70. There is no full discussion in the evidence of the code that gives rise to the displays in eRes. This problem is discussed further in Annexe 4 with reference to one screen (the incomplete reservation screen) only, which was considered by Dr Chiu in his third Report.

71. With a few exceptions where my own examination has revealed clear examples, I have not been provided with sufficient evidence to indicate that it is possible to recognise other layouts simply by reading the OpenRes and eRes code, or, if that is not the case, to indicate how much work is needed to extract the nature of the displays from the code. This is unsatisfactory, but the fact remains that all Navitaire's efforts were expended on demonstrating that the individual screens were either identical or similar, and that the association between copied screens and copied commands contributed to the case on 'non-textual copying'. The contention was that because the OpenRes code displayed the data in a particular layout with particular fixed data (column titles and so on) it recorded an artistic or literary work consisting of the screen layout: and because the fixed data and the layout appeared in the same place on the eRes screen, that artistic or literary work had necessarily been copied. It did not matter that the work could not be discerned from the code. The work was a design for an essentially transient object, a computer display; it was recorded in the software which, when executed, displayed it; it was therefore a copyright work, and any other similar screen copied from the first transient display infringed the copyright.

SUBSISTENCE OF COPYRIGHT AND INFRINGEMENT—THE LAW

72. While computer programs were protected as literary works under the Copyright Act 1956 at least by virtue of the Copyright (Computer Software) Amendment Act 1985, the Copyright, Designs and Patents Act 1988 contains express provisions to a similar effect. The 1988 Act was amended in a number of material respects by regulations made under the European Communities Act 1972 (the Copyright (Computer Programs) Regulations 1992, SI 1992/3233 and the Copyright and Rights in Databases Regulations 1997, SI 1997/3032). The 1992 Regulations do not refer expressly to Council directive (91/250/EEC) of 14 May 1991 on the Legal Protection

[10] Section (11) on page 16 of its proposed findings of fact.

of Computer Programs ('the Software Directive'), but must be intended to implement it, so far as UK law did not already do so; the 1997 Regulations expressly provide that they implement Council Directive No 96/9/EC of 11 March 1996 on the Legal Protection of Databases. It will be necessary to return to these Directives, because easyJet submit that they recognise and reflect a dichotomy between ideas on the one hand and the expression of ideas on the other that has not found much support in the English cases[11] thus far.

73. In summary, the issues are as follows. Navitaire contend that copyright subsists in the command set as a copyright work distinct from the source code. This claim has a number of aspects: (i) the collection of commands as a whole is entitled to copyright as a 'compilation'; (ii) each of the commands is a copyright work in its own right; (iii) alternatively, each of the 'complex' commands is a work in its own right. As to the displays, Navitaire contend that (i) in respect of the VT100 screen displays, the 'template' (fixed data and layout of variable data) is a separate copyright work for each display and (ii) certain GUI screens on the separate Schedule Maintenance module are copyright works as they stand and have been copied. Then it is said (and this is a quite distinct allegation) that the similarity exhibited by eRes to OpenRes in the eye of the user is such that there has been 'non-textual copying' of the whole of the source code. This is said to be strictly analogous to taking the plot of a book[12]: an author who takes the plot of another work and copies nothing else will still infringe copyright if a substantial part of the earlier author's work is represented by that plot, and the same goes for computer programs: *John Richardson Computers v Flanders* [1993] FSR 497 (Ferris J).

74. easyJet accept that copyright subsists in the source code of OpenRes. They submit, however, that since it is common ground that (so far as the user interface is concerned) none of that source code has been directly copied, the only question is whether a substantial part of that code has been taken. The only part of the code that can even arguably be said to appear in eRes is some (but not all) of the command names, and these do not amount to a substantial part of the code. Substantiality is to measured having regard to the skill and labour expended by the programmer on the choosing of those letters: and such skill and labour was trivial, although on any view the skill and labour in writing the software was substantial. They stigmatise the suggestion that copyright subsists in the command set as a compilation, or in the individual commands or some of them as an attempt by Navitaire to invent copyright works from aspects of the system that cannot be described as works at all. This goes for the VT100 displays as well, but the case on the GUI displays (which include the icons) is really only put as a matter of authorship and of substantiality. They contend that the case on 'non-textual copying' is objectionable because it extends the protection conferred by the copyright subsisting in computer software to matters that cannot legitimately be the concern of copyright, that is, to the intended effects of running the code on a machine *in the business sense*: there is no suggestion that the machine physically acts in the same way as the HP3000 machine running OpenRes—it is a quite different computer, involving different hardware etc, and its 'atomic' operations are quite different—but the results from the user's perspective are the

[11] See *Ibcos Computers Limited v Barclays Mercantile Highland Finance Limited* [1994] FSR 275, and Lord Hailsham's observation in *LB (Plastics)Limited v Swish Products Limited* [1979] RPC 551 at 629.
[12] *Harman Pictures NV v Osborne* [1967] 1 WLR 723, *Designers' Guild Limited v Russell Williams Textiles Limited* [2001] 1 WLR 2416.

same. Another way of putting the same point is that it is a claim to the functional idea of the program, rather than to the expression of that idea in software.

75. It is convenient to set out all the statutory provisions here.

Literary, dramatic and musical works

3.—(1) In this Part—

"literary work' means any work, other than a dramatic or musical work, which is written, spoken or sung, and accordingly includes—

(a) a table or compilation other than a database,

(b) a computer program,

(c) preparatory design material for a computer program and

(d) a database;

…

(2) Copyright does not subsist in a literary, dramatic or musical work unless and until it is recorded, in writing or otherwise; and references in this Part to the time at which such a work is made are to the time at which it is so recorded.

Databases

The acts restricted by copyright in a work

16.—(1) The owner of the copyright in a work has, in accordance with the following provisions of this chapter, the exclusive right to do the following acts in the United Kingdom—

(a) to copy the work (see section 17);

…

(e) to make an adaptation of the work or do any of the above in relation to an adaptation (see section 21);

and those acts are referred to in this part as the 'acts restricted by the copyright'.

(2) Copyright in a work is infringed by a person who without the licence of the copyright owner does, or authorises another to do, any of the acts restricted by the copyright.

(3) References in this Part to the doing of an act restricted by the copyright in a work are to the doing of it—

(a) in relation to the work as a whole or any substantial part of it, and

(b) either directly or indirectly;

and it is immaterial whether any intervening acts themselves infringe copyright.

…

Infringement of copyright by copying

17.—(1) The copying of the work is an act restricted by the copyright in every description of copyright work; and references in this Part to copying and copies shall be construed as follows.

(2) Copying in relation to a literary, dramatic, musical or artistic work means reproducing the work in any material form.

This includes storing the work in any medium by electronic means.

…

(6) Copying in relation to any description of work includes the making of copies which are transient or are incidental to some other use of the work.

…

Infringement by making adaptation or act done in relation to adaptation

21.—(1) The making of an adaptation of the work is an act restricted by the copyright in a literary, dramatic or musical work.

For this purpose an adaptation is made when it is recorded, in writing or otherwise.

(2) The doing of any of the acts specified in sections 17 to 20, or subsection (1) above, in relation to an adaptation of the work is also an act restricted by the copyright in a literary, dramatic or musical work.

For this purpose, it is immaterial whether the adaptation has been recorded, in writing or otherwise, at the time the acts is done.

(3) In this Part, ‘adaptation’—

(a) in relation to a literary work, other than a computer program or a database, or in relation to a dramatic work, means—

(i) a translation of the work;

(ii) a version of a dramatic work in which it is converted into a non-dramatic work, or, as the case may be, of a non-dramatic work in which it is converted into a dramatic work;

(iii) a version of the work in which the story or action is conveyed wholly or mainly by means of pictures in a form suitable for reproduction in a book, or in a newspaper, magazine or similar periodical;

(ab) in relation to a computer program, means an arrangement or altered version of the program or a translation of it;

(ac) in relation to a database, means an arrangement or altered version of the database or a translation of it;

…

(4) In relation to a computer program a ‘translation’ includes a version of the program in which it is converted into or out of a computer language or code or into a different computer language or code.

(5) No inference whall be drawn from this section as to what does or does not amount to copying a work.

76. A further provision has been added to the statute by reg 15 of the Copyright and Related Rights Regulations 2003:

50BA.—(1) It is not an infringement of copyright for a lawful user of a computer program to observe, study or test the functioning of the program in order to determine the ideas and principles which underlie any element of the program if he does so while performing any of the acts of loading, displaying, running, transmitting or storing the program which he is entitled to do.

(2) Where an act is permitted under this section, it is irrelevant whether or not there exists any term or condition in an agreement which purports to prohibit or restrict the act (such terms being, by virtue of section 296A, void).

77. The defendants submit that this is probably a belated implementation of Article 5(3) of the Software Directive, which seems to me to be correct, as does their contention that it probably doesn't matter, since s. 29 of the 1988 Act must anyway be construed conformably with that provision so far as possible.

THE WORKS RELIED ON

The complex commands

78. The manner in which the claimant's case has been put reflects the difficulties in identifying the proper approach to a command set such as the present one. The difficulties are (i) single letter commands, and those with longer names, even if clearly recorded in the source code, are unlikely to be entitled to a copyright—*Exxon Corp v Exxon Insurance Consultants International Ltd* [1982] RPC 69: (ii) whether it can be contended that the syntax of the 'complex' commands is recorded in the source code, and skill and labour went into their devising, and they are individual copyright works; and (iii) alternatively, whether it can be contended that the collection of command names as a whole is recorded in the source code and amounts to a compilation entitled to copyright even if the individual components of the compilation are not copyright works.

79. In my judgment, it is not possible to suggest that a copyright subsists in the individual command names as literary works. They do not have the necessary qualities of a literary work. The *Exxon* case wisely skirts the problem of providing a test for a literary work. There was no definition of literary work in the 1956 Act (section 48 merely stated that it included any written table or complation) and the definition in the 1988 Act is new. When one considers the modern definition (anything written spoken or sung which is not a dramatic or musical work—paragraph 75 above) it becomes essential to eschew any attempt at further definition. A single command name, or the word Exxon, is certainly written, and is plainly neither a musical nor a dramatic work. So why is it not a literary work? Laddie & al. *The Modern Law of Copyright and Designs* (3rd Edn) (hereinafter '*The Modern Law*') suggests that *Exxon* decides that the word is not a work, but warn that it is the composite phrase 'original literary work' which is what matters. There is obviously no bright line test. To attempt definitions *ad hoc* (such as, does it convey information or emotion?) is ultimately unhelpful. With great respect, this is particularly the case with old *dicta* from a different world, such as that of Davey LJ in *Hollinrake v Truswell* (1894) 3 Ch D 420, albeit that it was relied on by Stephenson LJ in the *Exxon* case:

> 'Now, a literary work is intended to afford either information and instruction, or pleasure, in the form of literary enjoyment. The sleeve chart before us gives no information or instruction. It does not add to the stock of human knowledge or give, and is not designed to give, any instruction by way of description or otherwise; and it certainly is not calculated to afford literary enjoyment or pleasure.'

80. In the 1988 Act, the phrase 'literary work' embraces tables or compilations, computer programs, preparatory design material for computer programs and databases. To concentrate on the word 'literary' may mislead, but it must not be ignored. In the end, the question is merely whether a written artefact is to be accorded the status of a copyright work having regard to the kind of skill and labour expended, the nature of

copyright protection and its underlying policy. It is not sufficient to say that the purpose of the act is to protect original skill and labour: there was plenty of that in *Exxon*. Nor is it of much weight that other forms of protection may be available. I think however, that it is clear that single words in isolation are not to be considered as literary works. The individual command words and letters do not qualify.

81. The second possible class arises from concentrating on the 'complex' commands alone. These are the commands that have a syntax, or, put another way, have one or more arguments that must be expressed in a particular way. Mr Arnold QC observes that it is not possible in fact to draw a sensible distinction between the commands identified by Dr Hunt as simple commands and those identified as complex. Dr Hunt described the classification of the DC command as a moot point, and there are others. I think that Dr Chiu's approach as he expanded on it under cross-examination is probably more satisfactory. It is necessary to distinguish commands that have one form from sets of commands with a common prefix. This is a workable distinction, provided it is remembered that the complexity of the effects of a command is not related to the complexity of its syntax. To describe a command as complex is just to describe its syntax, not its implementation nor its effects. It is possible to divide the commands up in this way. In such a classification, a command with two variants would be a complex command.

82. However, the division into simple and complex commands throws no light on the correct approach to the subsistence of copyright either in the collection of commands or in the commands considered individually. Some of the commands start little sub-parsers of their own: for example, the FEE command starts a series of prompts for further input, each line of which must be in an acceptable form. But the command itself is just the word 'FEE'.

83. In my judgment, the 'command word + syntax' approach to the complex commands in this case is not a valid one. I do not consider that the individual complex commands are distinct copyright works at all. The corresponding work cannot be identified. As pleaded, they are said to be literary works: that is, they must be written—see section 3(1). This aspect of the case turns, it seems to me, on whether and to what extent they have been recorded. They are recorded, in so far as they can be said to be recorded, in the manner I have described in paragraphs 36 and 37. In other words, the source code records them in the sense that it is possible to analyse the code to ascertain that a machine operating according to that code will 'recognise' the command A13JUNLTNAMS↵ as requiring the display of available seats on 13 June between Luton and Amsterdam. But this 'syntax' is recorded without being stated. The reason it is recorded rather than stated is that the reader, in effect, has to turn him- or herself into a machine in order to work out what the machine will recognise when operating according to this program.

84. This is a feature of all computer programs in what are often called procedural languages, which are the kind of languages with which this case is primarily concerned. It cannot be too strongly emphasised that a computer program controls a machine, and the result of that control may not appear from the program at all. Accordingly this part of the claim falls at the first hurdle.

85. However, I am acutely conscious that this may not be a satisfactory answer to the problem. It depends too much upon the way in which OpenRes was written. In answer to a question from me, it was made clear by Dr Hunt that it would be possible to

record the command names and their syntax expressly and use a program (a ‘parser generator’) automatically to construct a parser that recognised such commands accompanied by arguments according to such a syntax. The commands and their syntax would, in such a case, be recognisable as such in the source code for the parser generator. In such a case the copyright owner could point to a written work describing exactly how the alleged infringer’s program parsed the code and the consequences would be very different. I am most reluctant to come to a conclusion that depends upon the happenstance of the manner in which the programmer decided to set about constructing his parser. In either case, the ultimate result is a computer program which recognises the input according to the prescribed rules.

86. I consider that the better approach is to take the view that it is not possible to infringe the copyright that subsists either in the source code for a parser or in the source code for a parser generator by observing the behaviour of the final program and constructing another program to do the same thing. In expressing this view, I am verging on drawing a distinction between the ‘idea’ of the program and its ‘expression’, and, Navitaire contends, that is not a distinction known to English law and is entirely contrary to the observations of Jacob J (as he then was) in the leading case of *Ibcos Computers Limited v Barclays Mercantile Highland Finance Limited* [1994] FSR 275.

87. I think the problem should be approached in the following way. To define a series of commands and their syntax to be recognised by the computer is to define a computer language. It is exactly the same as defining a language such as BASIC or a simple language to control a calculator program. A program consists of a statement or series of statements in that ‘language’. Thus, to take the availability command as an example, one would say that the language includes an availability statement that starts with the letter ‘A’ and one of the permissible forms of which is A[date][City-Pair]↵. An example of a statement that will be parsed as an allowable statement to control the computer in accordance with this language is A13JUNLTNAMS↵. Recitals 13, 15 and 14[13] of the Software Directive are as follows:

> [13] Whereas, for the avoidance of doubt, it has to be made clear that only the expression of a computer program is protected and that ideas and principles which underlie any element of a program, including those which underlie its interfaces, are not protected by copyright under this Directive;
>
> [14] Whereas, in accordance with this principle of copyright, to the extent that logic, algorithms and programming languages comprise ideas and principles those ideas and principles are not protected under this Directive.
>
> [15] Whereas, in accordance with the legislation and jurisprudence of the Member States and the international copyright conventions, the expression of those ideas and principles is to be protected by copyright.

88. The Software Directive is a harmonizing measure. I must construe any implementing provision in accordance with it: if the implementing provision means what it should,

[13] They are not numbered in the original, but this is a convenient way to refer to them.

the Directive alone need be consulted: if it departs from the Directive, then the latter has been incorrectly transposed into UK law.[14] The recitals quoted are said by Laddie & al., *Modern Law of Copyright and Designs,* (3rd Edn) paragraph 34.19 to make it clear that 'computer languages are not included in the protection afforded to computer programs'. With this conclusion I agree, although the point cannot be said to be entirely clear and will require to be referred to the Court of Justice. In my view, the principle extends to ad hoc languages of the kind with which I am here concerned, that is, a defined user command interface. It does not matter how the 'language' of the interface is defined. It may be defined formally or it may be defined only by the code that recognises it. Either way, copyright does not subsist in it. This is of course not to suggest that the expression of a program in a particular language is not entitled to copyright. Quite the reverse. What this recital, and the associated dispositive provision of Article 1(2), appear to be intended to do, is to keep the language free for use, but not the ideas expressed in it:

> Art 1(2): Protection in accordance with this Directive shall apply to the expression in any form of a computer program. Ideas and principles which underlie any element of a computer program, including those which underlie its interfaces, are not protected by copyright under this Directive.

89. There is here more than an echo of a conceptual distinction between idea and expression, but it is unprofitable to pursue this approach in the light of the express reference to computer languages and interfaces in the recital and to the interfaces in Art 1(2).

The compilation of commands

90. The third possibility to which I refer in paragraph 78 above is to consider the collection of commands as a compilation of non-copyright items entitled as a whole to a copyright as a compilation. Mr Carr QC put considerable emphasis on two cases in particular. *Anderson v Lieber Code Co* [1917] 2 KB 469 is the case of the telegraphic code, and *Kalamazoo (Aust) Pty v Compact Business Systems Ltd* (1985) 5 IPR 213 is the case of the pre-ruled forms. *Anderson's* case is concerned with the Copyright Act 1911. The code consisted of 100,000 5-letter words that had been arrived at by generating 450,000 words, from which those that were unpronounceable and those that were likely to lead to an error in telegraphic transmission were eliminated. The 100,000 remaining words were used as a cipher. The judgment of Bailhache J in substance follows that of Kay J in *Ager v Collingridge* (1886) 2 Times LR 291, the core of his decision being that 'there could be no doubt of the enormous utility of the book, and of the service rendered to anybody desirous of forming a telegraphic code by an undertaking which once and for all eliminated words liable to error, and supplied such a collection of the aptest words for telegraphic use.' It followed that the collection of codes was the proper subject of copyright.

91. The *Kalamazoo* case concerned a collection of accounting forms which when used together made up an accounting system. Some of the forms were intended to be used in a peg-board system in which writing on the top form was reproduced on the lower forms in a stack, the forms being held in the correct register by a system of punched

[14] See Case C–106/89: *Marleasing* [1990] ECR I-4135, and in *Customs and Excise Commissioners v Century Life plc* [2001] STC 38.

holes, pegs and a clamp. Various collections of forms were sold by the plaintiffs, each collection being adapted for a particular purpose, and Thomas J in the Supreme Court of Queensland held that each collection or group of forms, designed to be used with each other, was entitled to protection as a compilation of the constituent forms even though the constituent forms were not wholly literary. Neither of these cases establishes any general proposition, save that (so far as the *Kalamazoo* case is concerned) if the quantum of skill and labour utilised in making a compilation of business forms themselves of a common nature is more than negligible, that is sufficient for copyright to subsist, although it may be noted that the judgment proceeds upon the footing that the line of US cases originating in *Baker v Selden* (1879) 101 US 99 is of application in Australia.[15] *Baker v Selden*, in which it was held that blank account books were not the subject of copyright, has given rise to the clear distinction drawn in the US between ideas and expression, and to its refusal to permit copyright to enter the field of the purely functional. For the reasons convincingly expressed by Jacob J in *Ibcos* (above) *Baker v Selden* is of limited utility in this jurisdiction, the development of the law of copyright having diverged from that in the US.

92. In my view, the answer to the compilation point is the same as that as I have given in respect of the individual commands. They are a computer language, not a program, and they should not be entitled to copyright. If I am wrong in this, however, I must consider Mr Arnold’s further submission, that there is no pre-existing material to form the subject matter of a compilation, and no compiler. His example is the NP commands, NP, NP. and NP-. Each is said to have been devised by Dave Evans and Mary Beesley, and he submits that there is no scope for selection or ordering. I do not think this is right. The commands are grouped by name into notepad commands. Such a grouping may involve a trivial exercise of skill and labour, but it is there. His better point, I think, is that there is no overall compilation, but merely an accretion of commands. The only influence that one command or set of commands has on the others is that the others must, by definition, have a different name. It is possible for a work that grows over time (say successive editions of Palgrave) to have a single compiler’s copyright, but there must be an overall design. I distinguish between the collection of commands needed for the system, upon each of which skill and labour was expended and the collection of their names, which was never part of the endeavour of the designers. The collection of command names and syntax was never designed as such. It did not have an author, and it did not have joint authors, since it is perfectly possible to distinguish the contributions of the various authors. On this ground also, I think that there is here no compilation.

93. I am aware that a great deal of interest has been excited by this question in the software industry. I was particularly interested in the development of the law in other countries, both within the EU and outside it, and the parties responded to this interest by providing a great deal of material from other jurisdictions. As I should have foreseen, the law is everywhere in a state of development, and the results differ from jurisdiction to jurisdiction. Within Europe, the German approach appears to be that identity of interface is not objectionable in itself, but may point to copying of the underlying code. In France, the user interface may be protectable. In the US, there are divergent authorities. In New Zealand and Australia, the point is undecided. While I have read much of this material with interest it has not, in the end, provided me with

[15] ‘It is well established that copyright is not available to forms of expression dictated solely by functional considerations, such as words which are merely part of an apparatus.’

assistance. The point has evidently vexed many judges and will no doubt vex many more.

94. Copyright protection for computer software is a given, but I do not feel that the courts should be astute to extend that protection into a region where only the functional effects of a program are in issue. There is a respectable case for saying that copyright is not, in general, concerned with functional effects, and there is some advantage in a bright line rule protecting only the claimant's embodiment of the function in software and not some superset of that software. The case is not truly analogous with the plot of a novel, because the plot is part of the work itself. The user interface is not part of the work itself. One could permute all the letters and other codes in the command names, and it would still work in the same way, and all that would be lost is a modest mnemonic advantage. To approach the problem in this way may at least be consistent with the distinction between idea and expression that finds its way into the Software Directive, but, of course, it draws the line between idea and expression in a particular place which some would say lies too far on the side of expression. I think, however, that such is the independence of the particular form of the actual codes used from the overall functioning of the software that it is legitimate to separate them in this way, and not to afford them separate protection when the underlying software is not even arguably copied.

The screens

95. The screens are obviously part of the user interface. The degree of similarity varies, and in my view the GUI screens raise different issues from those raised by the VT100 screens. There are 26 screens in all, and my analysis is set out in Annexe 5.

96. The VT100 is a character-based terminal, that is, it displays only printable characters. It provides 80 single-character columns and 24 rows for the display. As I endeavour to show in Annexe 5, one can see some of the layouts, at least, in the code because they are character-based, a good example being the baggage count display (item 11). The inference I draw from this is that the character-based displays are properly to be viewed as tables and so literary in character for the purposes of copyright (section 3(1)(a) of the 1988 Act above). They are, in my view, 'ideas which underlie its interfaces' in the sense used in Article 1(2) of the Directive: they provide the static framework for the display of the dynamic data which it is the task of the software to produce.

97. The GUI screens stand in a different position. The Directive is concerned only with the protection of computer programs as literary works, and I do not read it as having any impact on relevant artistic copyrights. It is certainly possible to view the GUI screens as tables, because they are 'drawn' by selecting from a palette of available objects things such as command buttons, toggle buttons, checkboxes, scrolling lists and so forth and moving them around on a form until a satisfactory layout is concerned. The 'interface builder' program provides 'stubs' for the routines that will be executed when the user selects or clicks on one of these objects, and it is the task of the programmer to provide the necessary code to ensure that the right thing happens when the user presses (for example) the OK button. Although composed of elements made available by the manufacturer of the interface builder program, I can see that the screen resulting from such an operation might properly be considered to be an artistic work. What the programmer ultimately produces is code that depends upon a large number of complex graphic routines that draw the background, the boxes

and the shading in the places selected, and act appropriately when the mouse moves over them or they are selected. The programmer does not write this code: it is the scaffolding for his or her own window design.

98. In my judgment, the better view is that the GUI screens are artistic works. They are recorded as such only in the complex code that displays them, but I think that this is strictly analogous to more simple digital representations of graphic works. The code constructs the screen from basic elements, and is so arranged to give a consistent appearance to the individual elements. I think, nonetheless, that to arrange a screen certainly affords the opportunity for the exercise of sufficient skill and labour for the result to amount to an artistic work. I consider that the GUI screens satisfy this requirement. There is force in the suggestion that they present a uniform appearance in layout of the elements, and so contribute to a uniformity of interface. On the whole this is sufficient skill and labour to entitle the screens sued on to artistic copyright.

99. In the result, therefore, the action fails so far as the VT100 screens are concerned, but succeeds so far as the GUI screens are concerned. The icons are plainly copyright works, albeit minor, and the action succeeds in respect of them as well.

THE REPORTS

100. Navitaire does not only complain of the copying of the screen displays during the operation of the reservation and check-in modules. They complain also of the copying of certain so-called reports. Reports are presentations of the data in the database, and are of particular importance in providing information about the functioning of the business. The reports are generated for the most part by individual programs. The evidence is not entirely clear as to how they are accessed, but it does not matter. My conclusions are in Annexe 6 below. In summary, this claim wholly fails.

THE 'HISTORY APPLICATION' OF OpenRes, AND COPYING IN eRes

101. OpenRes maintains an audit trail of all transactions. A record of every transaction is inserted into the History database. Part of the record written into the History database is a two-letter code identifying the nature of the transaction. Two commands, .H and .H*n* are used to query the History database, and produce a general history for the currently active booking and a detailed response respectively.

102. To refer to a 'History application' is a misnomer. There is no separate application. In OpenRes it consists of code that is invoked at many points in the program, and a dataset. The history codes (that is, the two character codes that are used to identify the nature of the entry in the database are specified in the COBOL code.

103. In eRes, there is also a separate history database. The names of some of the fields of the history table in this database are very similar to those of the corresponding OpenRes dataset. The two letter codes are not all the same, but very similar. The structure of this table was copied from OpenRes by taking the layout of the history data during the migration process. I will discuss this in detail when I consider the database claim, and it is not necessary to consider it here.

104. The software has nothing in common apart from the use of the same codes (there are sixteen) for identifying different history items. I do not understand copyright to be claimed in this set of codes. The two commands (.H and .H*n*) are the same. The screens are similar. The database representation is very similar. Thus (database issues

apart) the case is exactly the same as the case for business logic (see below) since a copyright is not claimed in the two .H commands.

105. The 'History Application' is no longer relied on as a copyright work, unsurprisingly since there is no identifiable module. It was abandoned as a distinct claim after service of the experts' reports in October 2003. It is thus part of the 'Business Logic' claim, but distinguished, as I understand it, by its standalone nature.

106. The History function is entirely banal. Dr Hunt described its corresponding database table as lying between the core tables and the pathetic ones (*sic*) because the functionality was necessary to a functioning system but not for the purpose either of effecting a reservation or of getting the passenger on to the aeroplane. She also accepted that *H or .H commands were common for history in reservation systems, that the function itself was commonplace, and that other systems produce displays similar to that in OpenRes. Two letter history codes are common, albeit that the eRes and OpenRes ones are very similar if not identical. It is accepted that the manner of operation of the two systems is different, but that they produce results that are consistent (the History dataset is one of the datasets migrated into eRes). Of course, there is no similarity in the underlying source code. This aspect of the functionality of eRes can only form an aspect of the non-textual copying claim and, for the foregoing reasons, is entitled to little weight.

THE OpenRes SYSTEM: "BUSINESS LOGIC" AND NON-TEXTUAL COPYING IN eRes

107. As I have indicated above, the case advanced by Navitaire is based on the fact that the functions of OpenRes and eRes are identical to the user so far as the aspects of the system of interest to easyJet are concerned. The case had its origin in the suggestion that what was called the 'business logic' of OpenRes had been appropriated.

108. During the course of her cross-examination on this subject, Dr Hunt gave this important answer.

> A. I think the problem with "Business Logic" and .... Perhaps we can just take a step back to where it came from. When I discussed this case with my instructing solicitors, we had drawn the lines that surround commands and screen layouts very tightly. Commands were, in my head anyway, really quite a limited feature. They are the text of the commands and what they do, but not in a detailed sense of what they do. Similarly, the screen layout is just the screen layout and very little else. It was obvious to me that there was something that has been copied in this case which is more than that. Now, I used the term "Business Logic". It may have been an error to use the term "Business Logic" and not pin it down more tightly at that stage, but it seemed to me that there is something which is the interaction between the commands and what you get and how you get the right data out at the right time in the process which I dubbed "Business Logic". "Business logic" does not have, like a lot of computer terms, a very precise meaning. It is not something that you would call a term of art. It does tend to be used.

109. As will appear from what I have said, I agree that the commands were really quite a limited feature. The question is whether the 'something else that has been copied' over and above the limited features of commands and screens is something Navitaire may protect from being copied. It seems to me that the following list sets out the matters copied over and above the comparatively limited aspects of the user interface:

    i) The relationship between the commands and the screens. No doubt this is obvious, but it is worth pointing out that the screens and commands do not exist in a vacuum. They are connected by invisible chains, and I do not think that in this context they should be considered separately.

    ii) The ability to carry out the operations of reservation, check-in, irregular operations and so on, with much the same commands and screens with a successful result.

    iii) Making the same data about all transactions as OpenRes provides available, and in substantially the same form.

    iv) At easyJet, at least, accordingly providing a 'drop-in' replacement for OpenRes.

110. This formulation concentrates on similarities and ignores differences. It is basic to the discussion that all the processing carried out by the systems, including background processing, is different: and it is accepted that the error processing, an essential feature of any program, is also different.

111. Dr Hunt's use of the term 'Business Logic' for what was taken was bound to run into problems, because it seems to assume that one can identify in the source code programming logic that in some way reflects the business logic. This assumption was explicit in Dr Hunt's original reference to pseudocode in section 3 of her April report, which was dedicated to the question of what business logic was. It is clear from her cross-examination that it was she who was responsible for the reference to pseudocode. This gave rise to a dispute with Dr Chiu that is enlightening.

112. I shall return to first principles. For present purposes, a computer running a particular program is a deterministic machine. A particular input to the machine will produce a predictable result derived from all previous inputs to the machine. If therefore one studies a machine in operation, it should be possible to identify the machine's response to all possible sequences of inputs, and so construct a new machine that operates to give the same outputs for the same sequences of inputs by writing an appropriate program. Navitaire contend that if this is done, it follows axiomatically that any copyright in the source code for the first machine must be infringed in writing the second program. Indeed, it was urged on me at an earlier hearing that it was unnecessary to consider any of the source code for the OpenRes system in determining whether there had been copying of a substantial part of the copyright(s) subsisting in the source code for it.

113. There is no doubt that easyJet and BulletProof had no access to the source code of OpenRes, and it is not in dispute that in languages used, actual code and architecture (subject to the claim in respect of the database) the systems are quite different. There is no suggestion that the eRes code represents a translation or adaptation of the OpenRes code. The term 'non-textual copying' might be replaced by the more accurate 'copying without access to the thing copied, directly or indirectly'.

114. The claim depends first upon the contention that the manner in which a machine behaves under the control of a program represents part of the skill and labour that went into the program. This is not an unreasonable observation. On the contrary, it is the whole object of the programmer to get the computer to behave in the required manner.

115. To copy an operating machine in this manner avoids the need to conduct any systems analysis or the production of functional specifications. Thus, it may be observed that although the copyist has not avoided the need to write software to achieve the desired result, he has avoided the need to identify the result by any of the normal methods of analysis that either precede or accompany the writing of a substantial piece of business software. Dr Hunt described this process as follows:

> Many different methodologies have been defined in the IT industry to help standardise and improve the way that these tasks are done and how the information they produce is presented. In some methodologies, such as SSADM (Structured Systems Design and Analysis Methodology) the assumption is that all details of a system will be documented, as a result of interviews with relevant client staff, before programming starts. In others, such as DSDM (Dynamic System Development Method) and MSF (Microsoft Solutions Framework) the assumption is that while the overall requirements will be set early on the details of how the system is to behave will be established in a series of iterative cycles, usually by building prototypes that are discussed with business users. It is also common, though not necessarily advisable, for developers to build systems without doing formal data analysis or business process analysis. This does not mean that the work involved is not done, just that it is done in parallel with coding.

116. In this connection, it may be noted in passing that no such formal analysis was ever carried out in the design of OpenRes. It is clear from the responses of David Evans, Greg McDaniel and Mike Padgen that they based OpenRes upon their previous experience and upon the comments and requests received from customers. It is not unfair to say that Navitaire object to BulletProof's and easyJet's acquiring the greater part of their experience from an examination of OpenRes in use alone. However, shorn of the specific user interface features that I have discussed, the operation of OpenRes cannot be distinguished from the manner in which other booking systems operate. Ms Antry, who was responsible for the design of the system was cross-examined on this point:

> Q. …You say this in the sentence bridging pages: "Most reservation systems support basic functionality including creation of new bookings, making changes to existing bookings, cancelling bookings, and other general reservations functions as required by the airline". I want to ask you a little bit about what is involved in this basic functionality that you are talking about there and specifically what is involved in creating a new booking. Do you understand? A. Mmm-hmm.

Q. It is common in the industry to have a command line interface, is it not? A. I do not have direct knowledge of that. I have worked with people in the industry who have used command line or screen mode or fill in the blank type of interfaces.

Q. But you are aware that quite a lot of systems out there have command lines interfaces. A. I am aware that there are a number of them, yes.

Q. What that means is that they will prompt the user to enter a command, parse the command once entered to see what sort of command it is and check the validity of the parameters entered. A. I do not know that that is the process they go through.

Q. Thinking about creating a new booking, the user will generally start by entering in an availability command to see what flights are available. Yes? A. That is a way it can be done, yes.

Q. The system will then search for flights that depart on the requested departure date and return on the requested return date or within a specified window? A. Yes, I would guess that it would search for the parameters input by the user.

Q. The matches will then be displayed to the user, typically in date order with a numeric index? A. I do not know exactly how the matches would be displayed, but the data will be returned to the user.

Q. Assuming that one of the flights is suitable for the customer who wishes to make the reservation, the user will enter a command to book so many seats in such and such a class on that flight. A. My understanding is, yes, they can request a number of seats.

Q. The system will then check to see if the flight selected has enough seats available in the requested class. A. My understanding, yes.

Q. If so, the system will reduce the number of seats available on the flight by the number of seats sold and store details of the booking in working storage. A. That is fairly accurate description of Open Skies, but based on my discussions with people at Sabre and WorldSpan, that does not sound familiar from what they have told me.

Q. Is that not something that any reservation system is going to need to do? A. Not in that specific time. My understanding from Sabre is that they do not store anything until all the data from the reservation has been entered.

> Q. I see. But certainly the system will need to reduce the number of seats available on the flight by the number of seats sold? A. At some point in time, yes.
>
> Q. At some point in time, indeed. Next the user will enter a series of commands to enter the passenger names and other required details. A. Yes, other information like they would request a price and enter the passenger information and payment information.
>
> Q. And again those would be added to working storage? A. Again, that is how Open Skies typically works, but I do not know. Like I have said, with Sabre, my understanding is that they wait and do that until they do the ER or the end record and transaction.
>
> Q. It would not be surprising if there were other systems that used working storage in that way? A. No. I am sure there are probably.
>
> Q. And typically there will be a running display of what has been entered and a prompt if any required information has been omitted? A. That could be probable.
>
> Q. Once all the required details have been entered, the user enters an end command and the system transfers all the booking details to main storage? A. That sounds reasonable, yes.
>
> Q. At that point you have a booking and the user can go on to some other task. You do not need to do anything more. A. As long as they have entered everything as far as selecting a price for the seat, because until they price it, the customer cannot fly on it.
>
> Q. Indeed, so one assumes that the seat has been priced at some stage in the process. A. Yes.

117. The overall functionality, from a business perspective, is what is accepted by Ms Antry in this passage. The steps are, in essence, check flights–check availability of seats–reserve–take passenger details–take payment details–record transaction. The seats thus sold must be made unavailable for future passengers, and that may be done at a number of stages in the transaction (including the initial grab: they can be returned to stock if the transaction is ultimately not proceeded with). This is common to reservation systems, and makes up what Dr Hunt calls the ‘core’ functions of the system.

118. The problem, therefore, if one is to arrive at a finding of infringement is to settle on a level of abstraction that describes something that is not merely inherent in the nature of the business function to be performed by the software, is taken by the defendants, represents the skill and labour of the designers and programmers but goes wider than the details of the command set and the screen displays, acknowledged by Dr Hunt to be a limited feature. Since copyright in computer programs is a literary copyright, the

natural approach for Navitaire is to base its contentions on the analogy between the function of a computer program and the plot of a literary work. Mr Carr QC employs the law in this area as a cornerstone of his submissions, and it is necessary to consider the cases.

119. In *Designer's Guild v Russell Williams Textiles Ltd* [2000] 1 WLR 2416, Lord Hoffman says at 2422:

> It is often said, as Morritt L.J. said in this case, that copyright subsists not in ideas but in the form in which the ideas are expressed. The distinction between expression and ideas finds a place in the Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) (O.J. 1994 L. 336, p. 213), to which the United Kingdom is a party (see article 9.2: "Copyright protection shall extend to expressions and not to ideas ..."). Nevertheless, it needs to be handled with care. What does it mean? As Lord Hailsham of St. Marylebone said in *L.B. (Plastics) Ltd v. Swish Products Ltd.* [1979] R.P.C. 551, 629, "it all depends on what you mean by 'ideas.'"
>
> Plainly there can be no copyright in an idea which is merely in the head, which has not been expressed in copyrightable form, as a literary, dramatic, musical or artistic work. But the distinction between ideas and expression cannot mean anything so trivial as that. On the other hand, every element in the expression of an artistic work (unless it got there by accident or compulsion) is the expression of an idea on the part of the author. It represents her choice to paint stripes rather than polka dots, flowers rather than tadpoles, use one colour and brush technique rather than another, and so on. The expression of these ideas is protected, both as a cumulative whole and also to the extent to which they form a "substantial part" of the work. Although the term "substantial part" might suggest a quantitative test, or at least the ability to identify some discrete part which, on quantitative or qualitative grounds, can be regarded as substantial, it is clear upon the authorities that neither is the correct test. *Ladbroke (Football) Ltd. v. William Hill (Football) Ltd.* [1964] 1 W.L.R. 273 establishes that substantiality depends upon quality rather than quantity (Lord Reid, at p. 276, Lord Evershed, at p. 283, Lord Hodson, at p. 288, Lord Pearce, at p. 293). And there are numerous authorities which show that the "part" which is regarded as substantial can be a feature or combination of features of the work, abstracted from it rather than forming a discrete part. That is what the judge found to have been copied in this case. Or to take another example, the original elements in the plot of a play or novel may be a substantial part, so that copyright may be infringed by a work which does not reproduce a single sentence of the original. If one asks what is being protected in such a case, it is difficult to give any answer except that it is an idea expressed in the copyright work.

120. This passage encapsulates the state of the law. Mr Carr QC submits that Lord Hoffmann is describing a case of 'non-textual copying'. That is no doubt so, if the meaning of those words is merely that the defendant has reproduced the copyright work in a different form, as in *Harman Pictures v Osborne* [1967] 1 WLR 723 where the similarity in language was slight but the choice of incidents the same, or in *Holland v Vivian van Damm Productions Ltd* [1936-45] MacG CC 69, where a ballet was held to infringe the copyright in a short story by Oscar Wilde. This principle extends to compilation cases: *Jarrold v Houlston* (1857) 3 K&J 708 is relied on by Mr Carr. This was a case in which the plaintiff had written a work which in the words of Page Wood V-C 'collects and reduces into the form of a systematic course of instruction those questions which he may find ordinary persons asking in reference to the common phenomena of life, with answers to those questions, and explanations of those phenomena.' He had provided answers to those questions out of works consulted by him and had arranged the whole 'under certain heads and in a scientific form'. Page Wood V-C expressed the principle thus:

> …if, knowing that a person whose work is protected by copyright has, with considerable labour, compiled from various sources a work in itself not original, but which he has digested and arranged, you, being minded to compile a work of a like description, instead of taking the pains of searching into all the common sources, and obtaining your subject matter from them, avail yourself of the labour of your predecessor, adopt his arrangements, adopt moreover the very questions he has asked, or adopt them with but a light degree of colourable imitation, and thus save yourself pains and labour by availing yourself of the pains and labour which he has employed, that I take to be illegitimate use.

121. Mr Carr QC says that this is a description of non-textual copying. That is not so as Houlston had taken the questions and the selection directly from the copyright work. But assume that he had not taken the questions, but merely the arrangement of and selection from the primary sources. That would still have been infringement: *Macmillan v Cooper* (1923) 93 LJPC 113, and that would be so even if the material abstracted from the prior works had been summarized or abstracted in different terms, provided that the selection had been copied.

122. What differentiates the present case from those to which I have referred above is that the copyist did not have access to the copyright work: in *Jarrold v Houlston* the copyist had access to the copyright work and used it. It may be noted that as he initially denied copying altogether it was subsequently difficult for him to identify convincingly the parts he had not copied: compare the *Ibcos* case (above). Two other cases are particularly relied on by Mr Carr. The first is *John Richardson Computers v Flanders* [1993] FSR 497 (Ferris J). This is a difficult case to summarise, but for present purposes I need only refer to a few salient facts. Mr Flanders, the defendant, had written the program the copyright in which was asserted against his new program. It was accepted that he did not have access to a copy of his earlier work when he wrote the later; and the judge rejected any contention of deliberate copying. He said this:

> In short, I do not accept the evidence I have discussed under heads (i) and (ii) in this section of my judgment as establishing

> deliberate copying of the [earlier] program by Mr Flanders. But the fact remains that he had, as I have said, an intimate knowledge of the [earlier] program at all levels of abstraction (to use the term employed in *Nichols v Universal Pictures Corporation* (1930) 45 F (2d) 119 and other United States authorities that I have mentioned) and *it is possible that he has, unconsciously or unintentionally or in some other way which he did not consider to be objectionable, made use of that knowledge* in a way that amounts to copying in the context of breach of copyright. It is that possibility that I must evaluate in appraising the particular similarities that I have identified. (my emphasis)

123. Seventeen similarities were relied on. Nearly all of them were rejected, but the ones which survived included the 'line editor'. Ferris J dealt with this feature in the following way.

> I find it impossible not to conclude that the line editor in the [later] program has substantially been copied from the line editor in the [earlier] program. If all that had been copied was the concept of a line editor that would not have mattered for present purposes, being a mere adoption of an idea. But similarities such as the obscuration of the text which is to be amended, the message "Insert to edit" in one case and "Copy to edit" in the other and, above all, the idiosyncratic restoration of text which is merely deleted and not replaced demonstrate that there has been copying of expression as well as idea.

124. The reference to 'copying of expression as well as idea' is a clear echo of the United States authorities discussed at length by Ferris J elsewhere in his judgment whose employment for this purpose was criticised by Jacob J in the *Ibcos* case, itself relied on by Mr Carr. But it is quite correct, as he submits, that it is possible to read the foregoing passage as stating that the concept of 'expression' as distinct from 'idea' extended to the manner in which the programmed machine worked, in detail. If this was what Ferris J was saying, then I would respectfully suggest that it is based upon a misapprehension as to the meaning of 'expression' in this context. But I do not need to go into this in detail, because I would with respect accept what is said by Jacob J in *Ibcos:*

> The true position is that where an "idea" is sufficiently general, then even if an original work embodies it, the mere taking of that idea will not infringe. But if the "idea" is detailed, then there may be infringement. It is a question of degree. The same applies whether the work is functional or not, and whether visual or literary. In the latter field the taking of a plot (i.e. the "idea") of a novel or play can certainly infringe—if that plot is a substantial part of the copyright work. As Judge Learned Hand said (speaking of the distinction between "idea" and

> "expression"):Nobody has ever been able to fix that boundary and nobody ever can."[16]

125. This does not answer the question with which I am confronted, which is peculiar, I believe, to computer programs. The reason it is a new problem is that two completely different computer programs can produce an identical result: not a result identical at some level of abstraction but identical at any level of abstraction. This is so even if the author of one has no access at all to the other but only to its results. The analogy with a plot is for this reason a poor one. It is a poor one for other reasons as well. To say these programs possess a plot is precisely like saying that the book of instructions for a booking clerk acting manually has a plot: but a book of instructions has no theme, no events, and does not have a narrative flow. Nor does a computer program, particularly one whose behaviour depends upon the history of its inputs in any given transaction. It does not have a plot, merely a series of pre-defined operations intended to achieve the desired result in response to the requests of the customer.

126. The view in favour of Navitaire's case is expressed concisely by the authors of *The Modern Law* in paragraph 34.64 (I have assumed that when they speak of 'obtains…from the original program' they do not mean obtain directly, but indirectly from watching the program work):

> For instance, the writing of a financing program may require as part of the task a careful elucidation of the relevant tax regulations—so that they may be reduced to a series of unambiguous statements—and it will be evident to any lawyer that this alone will probably involve a very large amount of work. A competitor might write a program of his own in a different computer language and arranged in a different way and with many improvements of his own but if he obtains the rules for calculating the tax from the original program instead of working these out for himself it is hard to see why he should not be considered a plagiarist.

127. There is a counter-example that throws some light on the nature of the problem. Take the example of a chef who invents a new pudding. After a lot of work he gets a satisfactory result, and thereafter his puddings are always made using his written recipe, undoubtedly a literary work. Along comes a competitor who likes the pudding and resolves to make it himself. Ultimately, after much culinary labour, he succeeds in emulating the earlier result, and he records his recipe. Is the later recipe an infringement of the earlier, as the end result, the plot and purpose of both (the pudding) is the same? I believe the answer is no.

128. I think that the answer to the problem is to be gathered from the passage in Lord Hoffmann's speech immediately following that quoted above (paragraph 119) from the *Designers' Guild* case:

> My Lords, if one examines the cases in which the distinction between ideas and the expression of ideas has been given effect, I think it will be found that they support two quite

[16] The quotation is from *Nichols v Universal Pictures Corporation* (1930) 45 F (2d) 119, and is set out by Ferris J in his judgment.

> distinct propositions. The first is that a copyright work may express certain ideas which are not protected because they have no connection with the literary, dramatic, musical or artistic nature of the work. It is on this ground that, for example, a literary work which describes a system or invention does not entitle the author to claim protection for his system or invention as such. The same is true of an inventive concept expressed in an artistic work. However striking or original it may be, others are (in the absence of patent protection) free to express it in works of their own: see *Kleeneze Ltd. v. D.R.G. (U.K.) Ltd.* [1984] F.S.R. 399. The other proposition is that certain ideas expressed by a copyright work may not be protected because, although they are ideas of a literary, dramatic or artistic nature, they are not original, or so commonplace as not to form a substantial part of the work. *Kenrick & Co. v. Lawrence & Co.* (1890) 25 Q.B.D. 99 is a well known example. It is on this ground that the mere notion of combining stripes and flowers would not have amounted to a substantial part of the plaintiff's work. At that level of abstraction, the idea, though expressed in the design, would not have represented sufficient of the author's skill and labour as to attract copyright protection.
>
> Generally speaking, in cases of artistic copyright, the more abstract and simple the copied idea, the less likely it is to constitute a substantial part. Originality, in the sense of the contribution of the author's skill and labour, tends to lie in the detail with which the basic idea is presented. Copyright law protects foxes better than hedgehogs. In this case, however, the elements which the judge found to have been copied went well beyond the banal and I think that the judge was amply justified in deciding that they formed a substantial part of the originality of the work.

129. The questions in the present case are both a lack of substantiality and the nature of the skill and labour to be protected. Navitaire’s computer program invites input in a manner excluded from copyright protection, outputs its results in a form excluded from copyright protection and creates a record of a reservation in the name of a particular passenger on a particular flight. What is left when the interface aspects of the case are disregarded is the business function of carrying out the transaction and creating the record, because none of the code was read or copied by the defendants. It is right that those responsible for devising OpenRes envisaged this as the end result for their program: but that is not relevant skill and labour. In my judgment, this claim for non-textual copying should fail.

130. I do not come to this conclusion with any regret. If it is the policy of the Software Directive to exclude both computer languages and the underlying ideas of the interfaces from protection, then it should not be possible to circumvent these exclusions by seeking to identify some overall function or functions that it is the sole purpose of the interface to invoke and relying on those instead. As a matter of policy also, it seems to me that to permit the ‘business logic’ of a program to attract protection through the literary copyright afforded to the program itself is an

unjustifiable extension of copyright protection into a field where I am far from satisfied that it is appropriate.

131. I am also confirmed in my view by the evident difficulty that the formulation of the 'non-textual copying' or 'business logic' case has caused the claimant. The claim was first set out by Dr Hunt in her April, May and June reports and it became clear that her approach to business logic involved creating an abstract view of the functioning of the software at many different levels, together with disregard for features of the systems that were in fact different. She concentrated on the user interface aspect, but ignored the background processing and error processing. By the end of the trial, the claim had undergone a further metamorphosis, being characterised as a claim in respect of the 'dynamic user interface'. Although a fine concatenation of buzzwords, this phrase is difficult to pin down as a matter of meaning. Although it was mentioned once by Mr Carr at the outset of the trial it appeared little (it is not used by Dr Hunt is her reports) until it was put at the centre of Navitaire's case in its submissions as to the material findings of fact. What Navitaire say is that they

> …rely upon the objective similarity between the eRes dynamic user interface and the OpenRes dynamic user interface as set out in the 2nd May 2003 and 24th June 2003 reports, [as confirmed by the Defendants' evidence] together with Dr Hunt's 3rd July 2003 Indications of Copying Report section 4 (for the dynamic user interface) and section 8 (for the History application).

I think that this is just a repetition of the case advanced on the user interface that I have dealt with. It adds nothing.

THE CLAIMS IN RESPECT OF TAKEFLIGHT

132. The claims in respect of TakeFlight are limited. TakeFlight is the program that provides the web interface to OpenRes. It is a 'web server' program, in that it sends pages of text in hypertext markup language (HTML) to browser programs running on computers in the homes of the customers. Embedded in these pages are small pieces of code in a language called JavaScript, which provide certain dynamic features of the page. TakeFlight interacts with the reservation system through an interface provided by the ORSVR01 module (Figure 1 above) through an API which is said to be not well defined.

133. Everything to do with TakeFlight is the concern of easyJet alone. While BulletProof defined and implemented the reservation system API, all the code for this allegedly infringing program was written in-house at easyJet by Mr Pritchard. BulletProof were not concerned with it. This is one of the advantages of a well-defined API: more than one author can write programs for it, as the huge range of programs written for the Microsoft Windows API attests.

134. It is common ground that TakeFlight was supplied to easyJet pursuant to the obligation to provide a web interface (paragraph 20 above). It was written by Justin Wilde, then a student working impossibly long hours as a programmer. The module is written in a language called Perl which is not compiled into object code but is 'interpreted' by a program called a Perl Interpreter. TakeFlight is the only piece of source code with which easyJet were ever supplied.

135. There are two aspects to the claim. One is a complicated complaint relating to what are alleged to be unauthorised alterations made to the TakeFlight code by easyJet over the years that it was in use. The other was a claim that the software now used by easyJet, referred to at trial for some reason as NIBS for 'New Internet Booking System', infringed the copyright in TakeFlight. The program is in fact called easyJet.com.

136. easyJet.com uses exactly the same API as the VT100 system to interface with the booking system. It also communicates by passing XML messages. So far as the outside world is concerned it looked rather similar to TakeFlight. There is not now any suggestion that the source code of TakeFlight was copied when NIBS was written, the contention having been abandoned in opening. A small piece of unused code remained in part of the site, but that is irrelevant. The complaint of infringement, therefore, is one of non-textual copying, and is based on the contention that the Perl code records a 'five-step' booking process which it is alleged was devised by Mr Wilde, and that to copy the 'five-step' booking process is to take a substantial part of the copyright code.

TakeFlight

137. TakeFlight is a program whose function is to serve pages to customers in a predefined order so that a booking transaction can be carried out. TakeFlight was integrated into easyJet's web site, which had been designed by a company called Tableau. This is often described as providing a 'skin' for an application, and Tableau had designed the 'skin' for TakeFlight. It was, so Mr Pritchard observed, a strongly easyJet look. The task of TakeFlight was to obtain the user's input; obtain the necessary data from the booking system; combine it with the fixed data forming part of the data for the Perl program, and transmit the resulting page to the user. It is a fairly complex application.

138. Dr Hunt describes the elements of the transaction thus:

> Step 1
>
> 7.5 Both NIBS and TakeFlight Step 1 pages contain the following functional items;
>
> - Departure location drop down list
> - Destination drop down list
> - Departure date, day and month drop down lists
> - Return date, day and month drop down lists
> - Number of passengers drop down list
> - Button to proceed to next page
>
> 7.6 The options for a customer to specify their return date include the option No, Just one way where the text "No," occupies the list box provided for the day of travel and the text "just one way" occupies the list box provided for the month and year of travel. This is a very unusual approach to the problem

of allowing both one way and return travel to be specified. A more typical approach would be to provide a separate 'radio button' selector to determine whether the trip is a return or one-way journey. This functionality is present in both systems.

Step 2

7.7 Both the NIBS and TakeFlight Step 2 pages contain the following functional items;

- Series of radio buttons to display and allow user to select outward flights
- Series of radio buttons to display and allow user to select return flights
- Button to proceed to next page

Step 3

7.8 Both the NIBS and TakeFlight Step 3 pages contain the following functional items;

- Details of selected outbound flight
- Details of selected return flight (if appropriate)
- Price details for selected outbound flight
- Price details for selected return flight (if appropriate)
- Option to proceed using extra security
- Option to proceed without using extra security

Step 4

7.9 Both the NIBS and TakeFlight Step 4 pages contain the following functional items;

- First name and last name of passenger with drop down list for title at the end
- First Name and Last Name of card holder
- Cardholder address, split into address and address continued boxes.
- Town/city
- Post code
- Telephone number

- Email
- Option to remember details for next visit
- Name on credit card
- Card number
- Expiry date
- Switch issue number
- Button to proceed

> Step 5
>
> 7.10 Step 5 of TakeFlight displays all the details of the user booking as confirmation. I have not been able to produce a NIBS version of this screen for this report as the site has changed since April 2002.

139. A number of points arise. Dr Hunt's inability to re-create step 5 means that she could not detect any differences in that stage, and there are some. Second, because these points arise in the 'similarities' report, Dr Hunt deliberately omitted points of difference she had detected when she first saw the interface.

140. Dr Hunt's Step 2 includes the display of available flights displayed according to the criteria stated in Step 1. Accordingly the five steps are a dialogue: ask for available flights–return available flights–select flight–display details and price of selected flight, inviting booking–book by giving personal and credit-card details–confirm booking. Navitaire contends (relying on Mr Wilde's evidence) that there is no need for step 3 if one is booking single leg flights. I cannot see why this is the case, and having read Mr Wilde's evidence I am none the better informed. In fact, the booking sequence mimics the sequence followed by a call centre agent. The cross-examination of Dr Hunt persuaded me that there was nothing in the claim for 'non-textual copying'. A sequence of obvious booking steps does not amount to a substantial part of the Perl code, which nobody even looked at. Dr Hunt spent some time analysing the various widgets used in the pages, such as scrolling lists, drop down lists, radio buttons and the like, but these are entirely conventional. Indeed, easyJet did not always use the same widgets as did TakeFlight. This claim should have been abandoned.

Unauthorised alterations to TakeFlight

141. A problem that was apparent to Mr Pritchard when he joined easyJet in 1999 was that of the need to modify TakeFlight to add new routes, or a new destination, or for short term promotions. easyJet also wanted foreign language versions, which were not available. The program suffered from bugs described by Mr Pritchard, and, apart from comments in the code itself, it was undocumented. It had no manual.

142. Mr Pritchard observes that a particular problem with TakeFlight lay in the fact that matters relating to presentation, such as language of the site, special offers and so on, were not separated from the program logic, as in a well designed application they

would be. This meant that the program logic had itself to be altered to accommodate what easyJet considered to be their requirements.

143. There is no doubt that easyJet copied the Perl code for the following purposes: (1) bug fixes (2) foreign language versions (3) adding and removing promotions (4) adding a new look and feel in the form of a new 'skin' and (5) using the program for easyJet Tours, which is a different legal person from easyJet. A degree of customisation to the program was permitted by an initialisation file called tflight_globals.pl. Mr Wilde exhibits this file, which encourages, but does not require, users to contact him to make the adjustments. I am not told in detail what degree of customization it permitted, but, whatever it was, it was not enough for easyJet's purposes. It is accepted that these modifications involve infringements of copyright unless for some reason Navitaire are debarred from alleging infringement. Two grounds are alleged: express agreement and acquiescence.

144. When easyJet started using TakeFlight, it was not clear that internet booking would be the way forward. From easyJet's perspective, it was important to be able to maintain a degree of control over the site to take advantage of new opportunities as they presented themselves. easyJet had always wanted to write their own site, and there were attempts to obtain a definition of the ORSVR01 API for that purpose. These were partially successful in that in either May or August 1998 Mr Padgen sent Mr Just part of the so-called 'buffer bible', the document used internally by Navitaire to document the ORSVR interface.

145. The functional modifications were mainly carried out by Mr Pritchard. He devised the 'Intelligent route selector', and produced the copies for promotions. In July 1999, he cloned the program to sell flights in French between Geneva and Barcelona. In November 1999 he introduced email booking confirmations. It is not necessary for present purposes to examine any changes he made after February 2000 since on 23 February 2000 Navitaire wrote to easyJet complaining of the modification of the TakeFlight source code. It is beyond doubt that easyJet were on notice after that date that further modifications were not approved. Thereafter, negotiations between the parties continued, but in the end they reached no satisfactory conclusion.

146. easyJet allege that there was from the outset an oral agreement that they should be free to modify TakeFlight. This allegation fails. There never was such an oral agreement. None of the witnesses ultimately supported it.

147. Mr Wilde knew that easyJet were modifying TakeFlight from June 1998. There is no doubt that by May 1999 Mr Wilde, who was a very unimpressive witness, and whose truthfulness I doubted, was aware not only that easyJet were making unauthorised alterations to the code, but had cloned the code to produce multiple running versions to run simultaneous promotions. There was a wide dispute whether there had been any complaint to easyJet about this prior to the disintegration of the relationship between Navitaire and easyJet in 2000. The person alleged to have complained is Mr Wilde. He could produce no emails complaining from before 1999 (his emails had been deleted from the mailserver) and after that time he said he said he sent no emails because he had become frustrated. easyJet produced over fifty emails from early 1998 to mid 1999 originating with Mr Wilde, none of which referred to this complaint. I find that no such complaint was made to easyJet. Indeed, Mr Wilde was assuming a good knowledge of the Perl code on Mr Just's part when he recruited the latter to a debugging exercise in May 1998. Mr Wilde suggested that he was 'authorising' Mr

Just to make changes in that area of the code, but this is a legalistic ex post facto rationalisation.

148. However, I find that Mr Wilde did nothing actively to encourage easyJet to modify the code, and I am satisfied that if he had told them not to he would have been disregarded. I think that easyJet were anxious to modify the code if it was necessary for a commercially successful operation of the site. Accordingly, there was no reliance by easyJet on any representation that Navitaire's rights would not be enforced, and no reliance by easyJet on Navitaire's failure to take any steps. To debar Navitaire from all relief there must at least reliance: see *Farmers' Build v Carier Bulk Materials* [1999] RPC 461. There is none here. So Navitiaire is not debarred by acquiescence or otherwise estopped from seeking relief in relation to the unauthorised modifications to TakeFlight.

149. In respect, therefore, of the claims for unauthorised alterations to TakeFlight, Navitaire are entitled to relief.

THE OPENRES SYSTEM: THE DATABASE CLAIMS.

150. There are a number of aspects to this claim. First is the question of reproduction of a part of the OpenRes database before the development of eRes proper took place, in what was called the 'Proof of Concept'. The Proof of Concept was a system designed to demonstrate to easyJet that BulletProof's suggested approach to the problem, that is a client/server structure using XML messaging and a database constructed using the Microsoft SQL server database management system, was capable of giving the throughput that easyJet required.

151. Next is the making of what Navitaire say are illicit copies of parts of the OpenRes database to enable BulletProof to investigate the contents and, Navitaire say, the structure of the database.

152. Then it is said that easyJet made illegitimate copies of the OpenRes databases for the purpose of 'migrating' easyJet's data from the old system to the new.

153. Finally it is suggested that eight tables in the eRes database reproduce a substantial part of certain copyright works, the database schemas or creation scripts, for datasets in the OpenRes database. Generally these can be referred to as the schemas, but they also form part of the database itself, where they are 'data about the data', or metadata. The programs called Adager and Suprtool can extract all the schema information from the metadata, and no doubt there are others as well.

154. Some technical understanding of the databases in this case is important to the analysis of the issues, because Dr Chiu maintained, and I accept, that certain technical features of the OpenRes databases mean that it was not possible to extract information as to the logical structure of the database with confidence from the databases themselves without access to the code that manipulated them: and it was common ground that this code was not accessible to the defendants. Dr Hunt had herself analysed the structure of the OpenRes and eRes databases in considerable detail, and presented a substantial body of evidence relating to similarities of structure, without, I have to say, clearly indicating in her reports whether or not those similarities were likely to be due to copying in the particular circumstances of the case. I used the materials supplied to me by the parties, supplemented by a substantial amount of instruction from the experts in the witness box.

155. In the discussion which follows, I refer to 'Source Safe' and to code being 'checked in'. Visual Source Safe is a Microsoft program that holds a database consisting of all programs 'checked in' to it. Every time that a program is checked out and then checked in, a record is kept of the alterations made to it. Using Source Safe, therefore, it is possible to obtain a history of the changes made to the program and of the dates those changes were made. Such dates must be used with care, since programmers differ in their assiduity in checking things in and out, and some source code may lie around for long periods without being checked in. Its utility in the present case was much reduced because many of the relevant records had been irrecoverably destroyed, but in some respects it was helpful.

156. The case also raises questions of difficulty concerning what the owner of the data in a database is entitled to know about his own data. It is convenient to consider the whole history before considering whether and to what extent the existing eRes system infringes any relevant work.

The works relied on

157. The claimants rely upon the following:

> 7.7 the literary work comprising the OpenRes Database structure set out in the database schemas (creation scripts)
>
> (a) as at 15th August 1994 (disclosed as document number 200 in the Claimant's List of Documents); and/or
>
> (b) relating to version 5.58 of OpenRes (disclosed as document number 210 in the Claimant's List of Documents);
>
> 7.8 the literary work subsisting in that part of the OpenRes database schema relating to version 5.58 of OpenRes which defines each of the tables within the OpenRes Database, and further or in the alternative each of the following tables namely:
>
> (1) AGENTS;
>
> (2) FLIGHT-FOLLOWING;
>
> (3) HISTORY;
>
> (4) QUEUE-RECORD
>
> (5) INVENTORY
>
> (6) SUB-TOPIC
>
> (7) TOPIC
>
> (8) AVAIL-FARES
>
> (A copy of the OpenRes database schema relating to version 5.58 showing the aspects thereof which define each of the OpenRes database tables was provided to the Defendants under

cover of letter dated 22nd October 2003 from the Claimant's solicitors).

158. Document 200 appears to be a set of schemas prepared for some purpose, but the presence of the extraneous numbers which would have prevented this file from acting as the source file for a database compiler means that it is not a document from which any database has ever been derived. I find that it probably does represent the state of the EAIS FLT database in 1994. In its submissions on the material findings of fact, Navitaire advance no case in respect of this document. It has never been formally abandoned, but its place in the history is not properly elucidated. I have checked one detail dataset in it (INVENTORY) which is rather different from the INVENTORY dataset in document 210. I will not consider this document further.

159. Document 210 is the transcript of a session using the database analysis tool Adager on the OpenRes database.[17] The tool is asked to output the database schema to the terminal, and this transcript is the result. It cannot be a copyright work on its own, as it was never used to create the databases. It is merely generated from the databases. The actual scripts used to create the databases may have existed, but they have not been produced. The reason I say they 'may have' existed is that it is possible to modify the databases without reconstructing them. Adager or some similar tool may also be used to specify the datasets in the first place, the result being recorded in a schema file. I accept that use of Adager in this way created a copyright work. Navitaire seek to treat the schema for each of the datasets as a separate copyright work, and I think this may be right. There are almost no relationships between the individual datasets expressed at the level of the database and thus each dataset may be said to stand alone, but the subdivision of data represented by the various datasets within the database is, of course, a matter of overall design. The relationships are maintained in the mind of the programmers who write the code that accesses the database and are ultimately implicitly expressed in the code. Against this, the fact that certain fields appear in more than one dataset reflects a degree of skill and labour in the design of the group of datasets as a whole. I can see no reason why the script for each dataset should not be considered an individual copyright work, or, alternatively, that the present state of the database records the metadata corresponding to a particular database structure.

160. Whether the notional scripts for the individual tables are individual works or the whole is a single copyright work seems to me to be quite irrelevant. There is a desire to minimise the size of the individual copyright works because of a concern that a substantial part of a 'small' copyright work may not be a substantial part of a larger work of which the small is but a part. Mr Arnold QC's submissions assume that this is the case. I do not think this can be right. Substantiality is not a question of proportion to the whole but of quality in context. As I have said elsewhere, the division of source code into modules and so on is as much a result of pressures extraneous to writing the software (such as debugging, maintenance and convenient building) as it is a result of deliberate design. Indeed, some methods of writing software may decide the modules for the programmer. I attach no importance to such divisions, although their appearance in allegedly copied code may be a thumbprint left by the copyist. To return to the database scripts, a banal table consisting of, say, a name, address and social security number may not be a substantial part of a large database work: but by the same token, it will be insufficiently original to be a copyright work considered on

[17] There is a more readable print of it exhibited to Mr Ronald Wallace's statement as RKW5.

its own. There is too much artificiality in copyright law as it is, and to contribute to it by looking for justifications for carving small works out of larger ones would be an undesirable development.

161. When the development of the eRes system started, easyJet's databases contained records of every reservation that had been effected using OpenRes. This was a large amount of data. The figure of 22 million records was mentioned. These historical records were, as I understand it, held in the live database, and this was a requirement from which easyJet were unwilling to depart. At all times, easyJet were anxious that the historical data was correctly 'migrated' to their new system, and this was understood by both parties to be a requirement of the contract. easyJet consulted their auditors on what was required for the migration: and one of the criteria was that eRes would have to produce exactly the same reports from the historical data as did OpenRes.

162. There is a comparatively limited number of persons who are central to this story. At BulletProof, Mr Vandertol was responsible for the overall development, and Mr Nuttall was responsible for the detailed design of the database. There are 497 stored procedures in the May 2002 version of eRes that perform all the tasks of accessing, reading from and writing to the database, and, as I understand the evidence, the great majority of them were written by Mr Nuttall. They amount to about 40,000 lines of code[18]. Mr Just at easyJet was responsible for liaison with the BulletProof developers. I did not regard Mr Vandertol as a reliable witness. He was vague and tended to ramble, but he was evidently knowledgeable. He often jumped to the first convenient explanation of what he was shown. From time to time, I think he did so to mislead. Mr Nuttall I did regard as a generally reliable witness. He was extremely familiar with the eRes database. Both were the subject of a sustained attack by Mr Carr QC, who was confronted with the not uncommon problem that the alleged infringement evidently involved a great amount of skill and labour. In order to provide a plausible motive for the copying alleged to have taken place it was forensically desirable to minimise the skill of the authors of the alleged infringement while credibly explaining their ability to construct a successful system that was in overall architecture, detailed design and implementation quite different from OpenRes. This attempt to reconcile the irreconcilable failed. That is not to say that there was not undoubted copying of the OpenRes database: there was, but, for the reasons I give below, that copying did not influence the design of the eRes database to a substantial extent.

The structure of the OpenRes database

163. The 'OpenRes database' consists of five distinct databases called CTL, FLT, VISADB, HISTDB and PEOPLE. The table creation scripts recorded in Doc 210 will generate (among others) five empty databases with these names. Of the datasets alleged to have been copied, none appear in the CTL database; six appear in the FLT database (INVENTORY, AVAIL-FARES, FLIGHT-FOLLOWING, QUEUE-RECORD, TOPIC and SUB-TOPIC) out of about 58 detail datasets; one in the HISTDB database (HISTORY) out of five detailed datasets and the AGENTS dataset from the 12 detail sets in the PEOPLE database.

[18] Not including blank lines. In the 'Visual Source Safe' that holds all the former versions of the code for eRes Dr Hunt said that there were thousands of stored procedures.

164. Navitaire contend that the database schema records (a) each table within OpenRes, each field within each table and the order in which the fields are arranged; and (b) information from which it is possible to draw an entity relationship diagram ('ERD') showing links between the tables in the database. Proposition (a) is true, although the order of fields is of no interest or significance save that similar ordering may be an indication of copying. Proposition (b) is so inaccurate as to be positively misleading.

165. Dr Chiu explained the nature of the datasets in a TurboImage database. There are three types of datasets: manual masters, automatic masters and detail datasets. OpenRes stores the attributes of the relevant entities (INVENTORY, for example) in detailed datasets. Many of these datasets are linked through keys (FLIGHT-SEG, for example) defined in automatic masters (FLIGHT-SEG-A) to other datasets (ten, in the case of FLIGHT-SEG). TurboImage does not permit direct relationships to be established between detail datasets. Automatic masters are in the nature of indexes, and they are constructed automatically. Detailed datasets cannot themselves be defined to contain unique keys. I set out the example given by Mr Padgen, on whose second witness statement the claimants rely in this connection:

[redacted]

166. The '!' before FLIGHT-KEY-A identifies FLIGHT-KEY as the primary key of this table. Thus, data is extracted from this table via an index stored in the FLIGHT-KEY-A automatic master. Now consider the INVENTORY dataset:

[redacted]

167. This is indexed on `FLIGHT-SEG`, a key maintained automatically in the automatic master FLIGHT-SEG-A. How does one know what the relationship between INVENTORY (what is available to be sold) and FLIGHT-STEP, which equally contains FLIGHT-SEG as a key? Mr Padgen says this:

> For example, the INVENTORY and FLIGHT-STEP datasets are linked by the FLIGHT-SEG field. Given an understanding of the names, it is obvious that although a particular FLIGHT-SEG would appear once only in the INVENTORY dataset, that FLIGHT-SEG would appear multiple times in the FLIGHT-STEP dataset as there are many possible FLIGHT-STEPS for a given FLIGHT-SEG. Hence it would be obvious to me for that link the 'one' would be into the INVENTORY table and the 'many' would be into the FLIGHT-STEP dataset.

168. The reason for the existence of FLIGHT-STEP and how it is used were described by Mr McDaniel, who designed them.

> 12. From late 1995, I also began work on the schedule/pricing part of the database: AVAIL-FARES, INVENTORY, FLIGHT-STEP, SCHEDULE and FARE CLASS-M. Between late 1995 and February 1996 I worked on the INVENTORY, FLIGHTSTEP and SCHEDULE, and then from February 1996 I worked on AVAIL-FARES and FARE-CLASS-M in collaboration with Lane Antry. These tables enabled the system to support through and connecting fares, both one-way and

> return, and allocate route-specific fares and link fares to availability so that the system would do all this in response to the reservation agent inputting the availability command. This was (and I believe still is) unique to the OpenRes system.
>
> 13. To support this additional functionality, the first stage was to create new datasets (AVAIL-FARES, SCHEDULE and FLIGHT-STEP) to enable through and connecting flights to be searched for. The system would search in SCHEDULE for all the schedules available starting at the origin and ending at the final destination cities that were inputted by the user. The SCHEDULE would recognise all of the different ways of getting from the origin to the final destination and, for each of these ways, would ask what flight steps made going from the origin to the final destination possible, and whether going non-stop was possible. This search was made possible by putting the FLIGHT-KEY field in both the SCHEDULE and FLIGHT-STEP datasets. Then, using the FLIGHT-SEG field which existed in both the FLIGHT-STEP and INVENTORY datasets, the system searched for the specific inventories of those segments.

169. I have no doubt that the analysis proposed by Mr Padgen is perfectly possible but it depends upon a thorough understanding of the function that the table is called upon to perform. Moreover, as Mr Padgen himself acknowledged, part of the analytical process would depend upon watching the database as data was written to it. For this reason, I accept that it is possible to deduce the relationships in the database but to say that the schema contains information from which it is possible to draw an entity-relationship diagram is part only of the truth. To draw such a diagram requires substantial effort. It needs confirmation from observation of the code.

170. Of course, what the schema does give is a list of the tables and their contents. Mr Wallace, who tried and failed to migrate the OpenRes data to eRes, acknowledged that. But his evidence did not confirm the evidence that Mr Padgen gave about the schema. Indeed, it rather confirmed my impression that a considerable amount of extraneous information was necessary before the OpenRes schema could be interpreted. It is accordingly necessary to deduce the structure of the database from the schema, using an intelligent assessment in the light of the business to be supported by the database of the records likely to be on the ends of one-to-many and many-to-one relationships. This is not a straightforward task. Dr Hunt and Dr Chiu both undertook it. Dr Hunt's evidence was that it was not easy, but certainly not impossible. It was not easy to see what she had thought had been copied. She was asked about this:

> Q. The conclusion that Dr. Chiu has drawn from all of this is that it would have been considerably easier for BulletProof to design their own database rather than attempt to copy the OpenRes database. That is right, is it not? A. For one thing, I have not said I do not think anywhere that they have slavishly copied this database as a whole. If you have got all the information or the user requirements and you have done all the data analysis, then it is surely better to design your own

> database. What the schema, once they got it, would have told them was that these are all the bits of data that we need to put into our buckets. That is a very open ended question to say is it easier or harder. You have got to add the detail in. Where are we starting from? Just to design from scratch a complete database is a big job.

171. The fact that the allegation of copying in eRes is only maintained in relation to a small proportion of the database must also be considered. Furthermore, the tables have very differing degrees of importance. TOPIC and SUB-TOPIC, for example, are part of the help system. Dr Hunt tends to use the word 'core' rather frequently, both when talking about the functions that the system has to perform and about the tables, and it is better to ignore it: sometimes I thought that everything identified by her as a similarity was 'core'. In each case also, the suggested manner in which the copying took place is or may be different.

172. For these reasons, I accept Dr Chiu's contention that to draw an ERD of the OpenRes diagram from the schema produced by Adager or Suprtool removing links which may be deduced to be unused, or to have a particular cardinality and without referring to the automatic masters, is not a fair view of the OpenRes schema.

Database structure in eRes

173. I think it is fair to describe eRes as an orthodox group of relational databases. It has a number of important features. First, and most important, is the ejWorkspace database. This database is designed to support the actual booking process, and it is used so that it is unnecessary to write to the ejReservations database until a particular booking is complete. To do this speeds the system up. The workspace database contains tables identical in structure to those in the main database: WrkContact, WrkReservation, WrkPersonName, wrkPayment and so on. Data is transferred from these tables to the corresponding tables in the main database only when a booking is complete. The workspace database is thus much smaller than the main database, and this greatly helps throughput.

174. The links and their nature (one to one, one to many) can in great part be ascertained from the schema. This is the major difference from the OpenRes schema, which in this respect is just a starting point for a lengthy investigation. The primary key for each table on the many end of a one-to-many relationship is a unique integer value generated by the so-called keyserver. Where there is a one-to-one relationship, of course, one table represents an entity and the other can be taken to represent further attributes of that entity, and the two tables share the primary key. The obvious example[19] is Flight and Flight-Following where Flight-Following contains the physical history of each flight defined by a row in Flight.

175. In addition to the data tables there are many stored procedures. These are stored in the database and are activated by the database server on receipt of the relevant message. There is no equivalent to the stored procedures in OpenRes. The necessary functions in OpenRes are carried out in COBOL code.

[19] It may be the only example.

What data to store in a database?

176. Before turning to the details of the history, I should refer to the manner in which database designers go about their work. In paragraph 14.8 of her September report, Dr Hunt said that

> The source of the information used to design the database is normally a combination of screen layouts, reports and business process analysis that are obtained during the analysis of user requirements. As discussed earlier BulletProof did not produce any substantial documentation of the proposed screen layouts, reports or business processes for eRes other than by detailing the existing layouts and processes of OpenRes.

177. I wish to concentrate on the first of these two sentences. When ‘user requirements’ are analysed, the product is a combination of the two computer-generated items (screen layouts and reports) from existing systems and one that is not—business process analysis. It is clear from the very extensive examination of the emails passing between BulletProof and easyJet that BulletProof’s approach was to work using the easyJet manual, the screen shots and the reports. There were many questions relating to the particular data: what was it for, and did easyJet use it? Dr Chiu, also, was in no doubt about the normal approach to systems design:

> MR. CARR: The assumption we are dealing with was that he was starting from a blank piece of paper. He sent these and said, "Look, use these in the design of your relational database." A. Yes, I mean, what tends to happen, it is a two-way thing, is that the developer will say, What data have you got? What information have you got in your stationery cupboard? Do you have any catalogues? We will go through this stuff. Do you have any existing systems? That is another common one. Do you have any existing screen print-outs? We will put all that into our data analysis exercise.

178. It is confirmed also by one at least of the textbook extracts with which I was supplied. Section 10.3 of *Database Systems: A practical approach to design, implementation and management* Connolly and Begg, (1995-2002) suggests that one of the sources of information for the designer of a new system includes ‘various types of flowcharts and diagrams; data dictionary; Database application design; program documentation; User/training manuals’.

The Development of eRes

179. It seems clear that the development of eRes was a consequence of a deterioration in the relationship between easyJet and Navitaire, largely contributed to by Navitaire’s reluctance or inability to provide easyJet with the API (applications programming interface) specification that would enable easyJet to write its own internet booking GUI for its website. Always a company that sold flights direct to the public, easyJet started by using a call centre. By 1997 it had 200 call centre staff and twenty aircraft, and the call centre was handling 6m calls per year. It saw in the Web the direction of its future expansion, and considered, therefore, that the web GUI was of primary

importance to it. Indeed, the evidence is that by 2001 90% of the reservations business was conducted using websites.

180. The development of eRes started with a call made by Mr Vandertol to the chief executive of easyJet, Ray Webster, in December 1998. Mr Webster, who is very experienced in the airline industry, has substantial programming experience. He wrote the open database connectivity ('ODBC') scripts enabling easyJet to access the OpenRes database to extract data for the easyJet reporting database, EZReportDB. He had known Mr Vandertol at Air New Zealand. Mr Vandertol had set up BulletProof in 1995, and appears to have been on the lookout for work.

181. The following account follows the references given in a helpful chronology prepared by Navitaire from both parties' submissions on fact. This document taken with the documents to which it refers is invaluable and generally accurate. Mr Vandertol had started in IT in 1985. In 1987, he joined Computer Solutions International, where he taught courses on dBase, rBase and Paradox database systems (none are relational). He also consulted with Air France, principally on issues relating to yield management. His exposure to airline booking systems at Air France was not substantial, as he himself volunteered in cross-examination. Mr Arnold says that Mr Vandertol was exposed to Air France's reservation system, but Mr Vandertol himself was not prepared to claim that he had derived much from that exposure, such as it was. From 1988-1990, he was employed by Ashton-Tate, manufacturers of the dBase software, and was responsible for writing technical documentation for the use of persons wishing to migrate databases using Ashton-Tate products to Microsoft SQL Server.

182. In 1990, Mr Vandertol went to Air New Zealand, where he worked for Mr Webster. He was a programmer, and responsible for such things as writing code to ensure network connectivity. He became acquainted with global distribution systems (reservation systems used by more than one airline, travel agents and so on) and with airline reservation systems. He wrote parts of a system for monitoring the sales activity of travel agents, called Telstar, which derived data from a number of sources including Air New Zealand's reservation system ('CARINA') and Sabre (a global distribution system). It retained four years' worth of financial information relating to 45,000 agents, and was a substantial system. He also devised a system for divergent itineraries which involved writing a relational database and the supporting code. Divergent itineraries arise when a single booking relates to different passengers with different destinations. eRes also supports them, although OpenRes does not, requiring instead the booking to be divided up.

183. In 1995, Mr Vandertol set up as a consultant under the name BulletProof Technologies. He developed TeleFlite, a system for providing real time flight arrival and departure information over the telephone using data from global distribution systems. He also customised TeleFlite for a company called DER Travel to quote tour prices and information, and developed a hotel reservation system for a company called Par Avion using a FoxPro database and the C++ programming language. Neither were particularly successful. In 1997, BulletProof was incorporated. BulletProof became a Microsoft Solutions Provider. Mr Vandertol worked on a very large system for Countryside Home Loans. This was on unreleased ('beta') Microsoft systems including SQL server. He was one of 300 programmers.

184. In about 1997, Mr Vandertol was approached by April Marenda, the CEO of a company called CIT Tours, who commissioned BulletProof to develop a reservations

system that BulletProof called DestinationPoint, and CIT Tours called resConnect. There are four diagrams from 1997 showing the analysis of the booking process in DestinationPoint. The project came to an end without a product in late 1998, but a pre-production version, with a database, was in existence. It too supported divergent itineraries. It was the work of Mr Vandertol, Avi Kleyman who had joined in 1997, Brian Milliron and Mike Mito. There is an entity-relationship diagram from September 1998 for this system. It was in essence a sales system, so it did not support check-in, irregular operations, schedule planning or more than one price per seat.

185. Mr Carr QC was anxious to demonstrate two things. One was that at this point, Mr Vandertol had no experience of ticketless booking systems. This was true. A ticketless system depends upon giving a reference number for a particular booking to the customer, and that booking being capable of being called up at the airport of departure and every airport along the itinerary from a central database. It is thus (and was at the outset) suitable for airlines with limited numbers of airports served and becomes suitable for large airlines only as reservation systems can be designed to be centralised, so that data is available in real time across the globe. To globalise access to a central database makes two other things possible: check-in systems can (and must, as there is no ticket with any passenger information) be integrated with the reservation system: and revenue information, and in particular how much each passenger has paid for a ticket, is available in real time to the airline. There is no doubt that ticketless systems are very advantageous when they can be used, but it was not clear to me how precisely the fact that the system was ticketless affected the data to be stored except in two respects, one general and one particular. The general matter is that it is possible to eliminate anything to support the production of printed tickets. The particular matter is that the pricing system can be moved from a per-route to a per-flight system, since the centralisation of the booking system enables all bookings on a particular flight to be priced in real time. Certainly, Mr Vandertol had no experience of booking systems for low-cost airlines before he undertook the development of eRes.

186. The second matter was that Mr Vandertol had no experience of writing code for large systems. This is not entirely right, since the Countrywide Loans system was on any view a large system but there were three hundred other programmers. In any event, to a certain extent the code is independent of the size of the system, but obviously it has to be robust enough to permit the database to become very large and still operate efficiently. But the main thrust of the point is correct. In a way, what I have to decide is how Mr Vandertol and the other programmers came up with a system that is so different internally from OpenRes and accepted to be so.

187. No doubt after the failure of DestinationPoint Mr Vandertol was anxious to find work. He telephoned Mr Webster in December 1998, and emailed BulletProof's business plan to Mr Webster on 18 December 1998. There is no record of any further contact until Mr Webster emailed Mr Vandertol on 12 April 1999 asking him to get in touch. Mr Webster asked Mr Vandertol if BulletProof could develop a reservation system using Microsoft SQL server that would be able to handle large transaction volumes. Mr Vandertol said that the only way to find out was to build a prototype, and easyJet accordingly commissioned BulletProof to construct a prototype. The prototype became known as the 'Proof of Concept".

188. Mr Vandertol visited easyJet on or about 22 June 1999 to discuss the proposal. At a board meeting of easyJet in July, it was agreed to go ahead with the proposal to fund a

prototype. Mr Webster felt some misgivings since he knew that IBM had tried to write a reservation system and failed, but it is clear that easyJet wanted a system over which it had control.

189. Before 19 July 1999, Mr Vandertol wrote a first draft requirements specification for the prototype entitled ‘easyJet EasyRes System: Usage Scenario Document’ (‘the Scenario Document’) which was discussed with Mr Just on the telephone on the 19 July. The agenda for the meeting was set out in the email (it is forwarded, as Mr Vandertol wrongly wrote out Mr Just’s email address on the first occasion):

1. Review of user scenarios for accuracy and completeness.

2. Review of existing OpenRes structure.

3. Initial work on Internet booking requirements for creating a new booking.

A computer program called NetMeeting was used to enable all the participants to consider the Scenario Document, and it seems as though BulletProof were enabled directly to access the OpenRes system by use of a program called Citrix.[20] Five screen shots were taken, and were incorporated into the next draft of the Scenario Document. The screen shots were the opening screen from the Flight Maintenance module, the Inventory Maintenance screen, the Fare Basis Code maintenance screen, two availability displays and finally the output screen from a program called CITYPFC0. I do not think CITYPFC0 was further discussed in evidence. It is the City Table Maintenance program and is used to alter the CITY-TABLE dataset in the FLT database. As it happens, this screen maps exactly the first ten fields in that dataset, but as it is not alleged to have been copied[21] it does not need to be considered further in this context.

190. The Inventory Record Maintenance screen is important. The INVENTORY dataset is one of those alleged to have been copied. It is instructive to compare this screen with the schema.

Screen	Dataset
Flight date	DATE-CTY-PR
City Pair	
Flight Number	FLIGHT-NO
Equipment	EQUIP
Seat config	
Lid	LID

[20] OpenRes has a VT100 interface. Most Windows computers have an application called Telnet, which emulates a VT100 terminal over a network connection. What easyJet and BulletProof did was to run OpenRes in a Telnet session in Luton, using Citrix to make that session available in California.
[21] Nearest equivalent is the AirportCode table in the ejFlight database.

Screen	Dataset
Capacity	CAPACITY
Inventory trigger	INV-TRIGGER
Service Flag	
Airline Code	AIRLINE-CODE
Arrival Time	ARRV-TIME
Sold non-stop	SOLD-NS
Sold Connecting	SOLD-CONN
Sequence number	
Departure time	DEPT-TIME
On Time	ON-TIME
Sold	SOLD
Sold thru	SOLD-THRU
Count number	COUNT-NUM

191. In addition to the foregoing fields in the schema, there are six Cabin Fares items reported in the screen display, each of which is present in the schema in the form of three arrays of six members, CABIN-CAPACITY, CABIN-LID and CABIN-SOLD. The maintenance screen, which must be used by the airline every time it specifies a flight on which seats are to be sold (i.e. the inventory to be sold) thus sets out all the variables that the system uses to describe it.

192. Important too is the Fare Maintenance and Pricing screen. I will not set out another table comparing them, but this screen sets out substantially the whole contents of the FARE-CLASS-M dataset.

193. On 19 July 1999 also, a login was created for Mr Vandertol to enable him to access the user interface of OpenRes directly. There is no satisfactory evidence of this logon having been used, and it seems as though there was little or no direct access of the OpenRes system from BulletProof until February 2000, when a Citrix logon (or at least a logon permitting BulletProof to examine the contents of the database directly) was created and was used by Mr Vandertol to some extent (not as much as 10 times or 20 times) between February and April 2000. (I think that use of this logon is probably responsible for the unused Fares.tbl and FareClass.tbl in the Tables folder of the Proof of Concept source code: paragraph 202 below.)

194. The second draft of the Scenario Document (version 1.1) was emailed to easyJet on 22 July for a conference call to take place on the same day. Along with the screens are a series of questions about the use of the data to which they refer. In version 1.1 of the Scenario Document, a table rather like the one I have set out above is set out immediately after the Inventory Record Maintenance screen shot. Against each data item in the screen display are two columns, headed respectively ‘Used’ and ‘Required’. The ‘Used’ column contains information on the use made of the data item by easyJet: the ‘Required’ column identifies only a few lines. The table is followed by

comments and questions relating to the data. The same is done for the Fare Basis Code Maintenance screen, which sets out the content of the FARE-CLASS-M dataset.

195. I should observe at this point that BulletProof are here investigating the OpenRes system in precisely the manner described by Dr Hunt and by Dr Chiu, to which I have referred above. They are plainly engaged in finding out what data exists about the business and system used by their client.

196. The next version of the Scenario Document is the result of a further conference call on 29 July 1999. Version 1.2 is called 'Requirements Fare Pricing'. It is described as 'Version 1.2 amended by J. Vandertol included comments from John and Clive referenced conference call on Thursday July 29 @ 8 am local time'. It now only has four screen shots. The tables following the Inventory Record Maintenance screen and the Fare Basis Code Maintenance screen have been shortened, the former to some only of the data items (including those described as 'required') in the previous draft (paragraph 194 above). It is preceded by the words 'From the table above the following data points are required.' The table below the Fare Basis Maintenance screen-shot describes only six fields. A discussion of association of the fare value with the flight in the existing system follows, and is described as working satisfactorily but as requiring some changes. The decision is taken to move tax rules out of the fare basis area. This is a design decision which appears to have been followed in eRes.

197. The document concludes with the 'Transaction Requirements'. These specify the number of transactions of various types that the system must be capable of supporting. The Internet and Call Centre requirements are specified separately.

198. After two minor revisions, version 2.0 of the Scenario Document (again renamed, this time to 'Prototype Requirements Document' was produced by Mr Vandertol on 1 September and version 2.1 on 28 September. This now includes screenshots of easyBook (that is, easyJet's version of the TakeFlight website). Version 2.5, the final version, describes the transaction requirements in some detail, and describes the skeletal website that will be used to test database throughput.

199. From about September onwards, the database for the Proof of Concept was being designed. Entity-relationship diagrams for this database exist: EZjetFare.er1 was checked into Source Safe on 5 October. On 13 October 1999, Mr Vandertol emailed a third draft of his schema. Navitaire put a great deal of weight on the contents of this email, which I should quote:

> I spoke with John on Monday this week and we [made] some changes to the schema. We have since been working with the schema and believe that additional work, in this case simplification may be in order with the route segment and fares area of the schema. This design was intended to support through fares efficiently. At this point though, *I am not fully convinced that a simple table similar to what is in OpenRes might not do the trick as well.* Additionally during the conversation with John we added fields to qualify the fares more closely. We also identified that certain characteristics that affect the availability of the fares would be handled by the yield management system. The yield management system would deal

> with all aspects of fare availability except those that cannot be determined until booking time. The booking system should support simple attributes of a fare that are provided or identified during the time of booking. However more complex attributes should not be placed in the reservation system. We draw the line of complexity when a fare rule would require that the itinerary be reviewed to see if it qualifies. Though we added in attributes such as length of stay, this rule would require a review of the itinerary to qualify it. Therefore careful consideration is required if it should stay in the table. Finally we need to look at the behavior that will be supported when a price boundary is crossed.
>
> I would like to discuss these issues as well as the thru fare issue during todays call. Please find a gif of the schema attached. [my emphasis]

200. In the schema annexed to the email the CabinFare table is obviously closely related to OpenRes's FARE-CLASS-M, the maintenance screen for which was so extensively discussed in the Scenario Document. It seems as though it has had fields added to it, and certain parts transferred into the FareBasis and FareType tables. It was put to Mr Vandertol that the 'simple table' he was talking about was the OpenRes AVAIL-FARES, which has 137 fields if the array fields are expanded. It is implausible that AVAIL-FARES is the table to which Mr Vandertol is referring, the more so since its essential features are concerned with fare availability (those fares that are active, to be sold and sold) and this has no echo in the CabinFare table in the Proof of Concept database. I do not accept, therefore, that Mr Vandertol had any knowledge of the AVAIL-FARES table at this stage of the development. I find that the CabinFare table had its origins in FARE-CLASS-M through the maintenance screens for the Scenario Document, and this is the table to which he is referring. In coming to this conclusion, I have taken into account Mr Vandertol's denials: but the maintenance screen was plainly used as a source for ascertaining the necessary data.

201. Mr Vandertol altered the EZJetFares.er1 again and the new version was dated 15 October 1999. The role which this schema had in the development of the Proof of Concept is not entirely clear, but its version of the CabinFare table is the same as that in the Proof of Concept. The Proof of Concept database was finished by 16 December 1999, when it was made accessible to easyJet on the Internet. The interfaces were completed by early February 2000, and the whole was demonstrated in California on 8 February 2000. It was successful.

202. The database scripts for the Proof of Concept database (which appear to have been checked into Source Safe in April 2000) include 11 additional table scripts, two of which are SQL-server versions of the OpenRes scripts for the AVAIL-FARES and FARE-CLASS-M tables. They are called Fares.tbl and FareClass.tbl respectively.

203. The presence of the Fares.tbl and FareClass.tbl scripts was never satisfactorily explained. While they were checked into the same directory in Source Safe as the Proof of Concept table scripts, they are not in any way linked with any of the Proof of Concept tables, which themselves do not contain any keys referencing these two tables. This is a fact that Dr Hunt should have referred to in her reports, as she recognised. They have not been edited to conform to what appears to be BulletProof's

standard form of table, with a unique table identifier as the first field (column). Two of the columns in FareClass.tbl have names (BaseFare, ChangeFee) that are the same as those of columns in one of the actual tables (FareBasis.tbl), but subject to that no name of any column is referred to, as far as I can see, in any of the stored procedures in the Proof of Concept. Navitaire are emphatic that these tables demonstrate that the Proof of Concept was developed from OpenRes. In my judgment they show no such thing. To show that such a development took place, it should be possible to demonstrate something that has been taken from these two tables and finds its way into the Proof of Concept, but it is not possible. The defendants are right when they say that these tables played no part in the Proof of Concept system. However, I am quite satisfied that the tables represent the result of having had either direct access to OpenRes before the date in April 2000 when they were checked in to source safe, probably by the use of Adager or a similar tool, or access to the database over an ODBC link. Whether that access took place before October 1999, when the structure of the CabinFare table for the Proof of Concept was substantially finished is quite uncertain.

204. A great deal was made of these tables by Navitaire. I think that much of the forensic excitement at the time was generated by the fact that it had not been observed that the scripts were just present in the Tables directory in the Proof of Concept in the Visual Source Safe, having been checked in in April with everything else. In their submissions on the facts, Navitaire contend that

> Whilst it is true that the two tables were not linked to the tables in the Proof of Concept (as at April 2000), there is absolutely no doubt that copies had been supplied to BulletProof. Use of Proof of Concept had finished by February 2000 when it had been successfully tested and thus there was no reason for Proof of Concept to have been altered after this date. So they must have been there by February 2000.

205. This is full of misapprehensions. There are compelling reasons for supposing that they were never linked to the tables in the Proof of Concept database: I have given the principal reasons above. Next, to check in a table script in a particular directory merely means that it was checked in: it says nothing about where it was or how old it was when it was checked in. Reference has to be made to the scripts for building the whole system to see if they refer to these tables, and they don’t. Mr Nuttall tended to check things in in bursts, and, as Mr Vandertol put it, they may just have been lying around. I have no difficulty in accepting this explanation.

206. The Proof of Concept system had certain of the principal architectural features of eRes. The database management system was, obviously SQL Server. The API was defined in the same way as it is in eRes, by the use of XML messaging. The tables, as I have indicated, were defined in the house style, and the supporting software in the VT100 software was written in Visual Basic.

The Flight table in Proof of Concept

207. The Flight table in the proof of concept database is plainly directly related to its predecessor in EZJetFare and in version 2.5 of the Scenario Document. Indeed, apart from KiloDistance, which does not have any correspondence with any data item in OpenRes, all the data items in this table are directly derived from these two sources.

208. The next event is the contract for eRes and the development of eRes, and it is convenient to assess what BulletProof in general and Mr Vandertol in particular knew of OpenRes. Dr Hunt, who was quite conscious of the fact that the contents of the FARE-CLASS-M dataset was available through the maintenance screen, said this in paragraph 14.32 of her September report:

> It is clear from my examination of the documents relating to the Performance Prototype in section 5 and in the Proof of Concept database section above, that Joseph Vandertol in particular was aware of the internal structures of OpenRes well before development of eRes commenced.

209. This statement cannot be supported. It is absolutely correct that Mr Vandertol was well aware of what was in the maintenance screens that I have discussed. They contain the column names of two datasets in the database, not its structure. If it is appreciated that each maintenance screen is concerned with one dataset in the database, there is no doubt that the maintenance screens give a good idea of the contents of the individual datasets and to that extent some clue as to the structure of the database but there is no information whatever about the interrelationships or links between the tables. Dr Hunt accepted in cross-examination

> The basis is that the screens that he had seen were the ones that talk directly to the tables in OpenRes. I think saying internal structures is perhaps overstating it a bit here.

Dr Hunt did indeed, on occasion, overstate things a bit. The problem with the allegation contained in her report is that it was central to Navitaire's case and much investigation is needed to ascertain its full scope. Unfortunately, a proper analysis shows that there was no material in Mr Vandertol's possession from which it was possible to ascertain the 'internal structures' of the OpenRes database. It is convenient to mention here one further example of her approach which gives me great concern. For the purpose of considering the relationship between the tables in eRes and certain of the tables in the DestinationPoint database Dr Hunt consciously applied a different criterion from that she applied when she compared the tables in eRes to those in OpenRes. She applied different criteria for similarity because, she said, DestinationPoint names could be copied physically, and so any difference in name represented a real difference. She said this:

> A. Can I stop you right there because you are proceeding on an assumption as to what I have done in this comparison being the same as the comparison of OpenRes and eRes and that is not right. The statement that was made that I was considering in here was we started with DestinationPoint. DestinationPoint ERD is held in a product that allows you to dump the SQL Server schema out of it. So my basic assumption was that if you were going to have a first name and it came from DestinationPoint, it would look exactly like the first name in DestinationPoint, whereas my assumption with OpenRes was that there was bound to be some difference because you have to transcribe the thing; but you cannot automatically do it. I want to set that in context because it is a slightly different exercise. If you apply the same rules, the answer will be different, but I was

> not doing exactly the same exercise here. I was looking to see if this was actually the physical starting point for eRes.

210. In fact, it is part of Navitaire's case that BulletProof did indeed have access electronically to the OpenRes tables, and, as she accepted, it is possible to 'automatically do it'. Dr Chiu's criticism, that she tended to underestimate the similarities between DestinationPoint and eRes, but exaggerate those between OpenRes and eRes, is justified. As she said, if you apply the same rules, the result will be different. So the real process of comparison that she undertook is completely unexplained in her report, even though the results are contained in similar tables with red, green and orange highlighting to represent no copying, some copying and a copy. In general, Dr Hunt's reports are excessively conclusory, and while much of the purely factual material which is presented is accurate, there are omissions[22] and a great deal of the underlying material is not explained to the reader. Her conclusions are not necessarily reliable, and I have found my labour greatly increased by the need constantly to refer to widely disparate sources in consequence.

The eRes contract

211. Negotiations appear to have begun in February 2000, at about the time that Mr Webster and Mr Just visited BulletProof to see the Proof of Concept system. Clearly, BulletProof were anxious to show that the requirements for the test were met, and Mr Vandertol explained that some code from DestinationPoint was used to run the tests. The evidence is that initially easyJet were interested in contracting with BulletProof for the database and API only. This would have enabled easyJet to construct their own web GUI and to construct whatever call centre and check-in interfaces they required, but ultimately BulletProof was asked to develop the VT100 interface for the call centres and check-in terminals as well. BulletProof started work on the system before the contract was signed.

212. The matter was discussed by the board of easyJet at their June 2000 meeting. A minute records that the investigation of the feasibility of a replacement system running under Windows NT was well under way, and that the first prototype completed in January proved that the software/hardware platforms selected would exceed the specified demand, which was stated to be thirty times the largest newspaper promotion in 1999. A decision appears to have requested. Perhaps the underlying reason for the whole development is stated in this comment on the status of the OpenRes system: 'all development is on FlightSpeed and the new internet development system Skylights (which incurs a charge for each transaction handled!)'.

213. The brief to BulletProof included the requirement that with minimal changes and improvements, the call-centre and check-in VT100 interfaces were to be unchanged, Schedule 1 to the Agreement provides a short specification (this is the whole of it, albeit not final):

> The Software should give the same facilities/provide the same functions as easyJet's present OpenRes system but with certain additions: ability to change internet bookings and flexible Passenger Name Records ("PNR's") or other such name

[22] Including some examples of selective quotation of error messages in the May report which have the effect of making the messages rather more similar than in fact they are.

> systems. Further with respect to performance, the system must be capable of handling 30 times the number of transactions as easyJet's present OpenRes system for a specified promotion.

214. The contract contains an indemnity from BulletProof to easyJet against copyright infringement, except in respect of claims by Hewlett-Packard in respect of the user interface. It is not necessary for my purposes further to consider the contract between BulletProof and easyJet.

215. There is one other matter. It was always an easyJet requirement that the relevant data in the existing system would have to be transferred to the new system. This process is called migration, and the migration of the data in this case not only gives rise to a distinct claim of infringement of copyright but is intimately tied up with the suggestion that the tables complained of in eRes are copied from the OpenRes database.

Development of eRes: access to OpenRes

216. Dr Hunt expressed the view that eRes specifications were thin, and the cornerstone of Navitaire's case is that BulletProof had unrestricted access not only to the user interface of OpenRes but also to its accessible internals in the form of the database table scripts in the form that is produced by tools such as Adager or Suprtool. BulletProof is said to have been further assisted by direct access to the database provided by direct links to it contained within a Microsoft Access database called bp.mdb. The sequence of events is as follows.

217. In February 2000, development of the database structure started. Not only did it not spring into existence, it took a considerable time. As I have mentioned, on 11 February 2000 easyJet created a login for BulletProof to enable the latter to access OpenRes. Shortly thereafter, this login was given supervisor privileges enabling the user to access the system maintenance functions. This logon was used regularly, and I imagine that it was used extensively for the purpose of investigating the user interface. BulletProof was also provided with copies of the OpenRes training manual, and the manual used by easyJet for training staff on making reservations. I think that it is probable also that this logon was used to investigate the database.

218. On 18 June 1999, John Rybka at easyJet had created a spreadsheet called OpenResStructureMaster.xls, containing information about the OpenRes database. His evidence, not successfully challenged, was that he thought he had produced it at the same time as an Access database called TableDefinitions.mdb, for the purpose of creating a data dictionary (document 210 is a data dictionary) for constructing reports. It is necessary to explain that easyJet maintained a number of systems that needed access to the database. One of the datasets (FLIGHT-FOLLOWING) is largely populated by data inserted by routines written by easyJet. Dr Hunt identified these sources:

> easyJet have said that many of the similarities between the eRes and OpenRes databases are due to the existence of external interfaces, built by easyJet, that accessed OpenRes for various reporting functions. I have reviewed the available information on these interfaces, which includes some source code. The diagram below shows the main components that I believe

existed before the development of eRes, although the way that easyJet refer to these interfaces changes over time and is somewhat confusing. I have used the name ‘ezReportDB’ to refer to the reporting database, it is also referred to as ‘ezDB’ and possible ‘easySQL’. The diagram shows three main components;

EzMVT	Uses external messages to update FLIGHT-FOLLOWING in OpenRes
EzRMS	Reads AVAIL-FARES and INVENTORY, processes revenue data and then updates INVENTORY
EzASM	Reads from ‘ezReportDB’ and updates FLIGHT-FOLLOWING
EzMinute	Reads from INVENTORY, FLIGHT-RELEASE and SCHEDULE and inserts data into ‘ezReportDB’
ezOvernight	Reads from SEGMENT, PNR-DETAIL,PNR-ADDRESS, CURRENCY-TABLE and inserts data into ‘ezReportDB’

219. Dr Hunt says, and Dr Chiu agrees, that whoever wrote these interfaces had a good understanding of the dataset structure of OpenRes. Each communicates directly with the database over an ODBC connection, and thus can write without any of the validation checks enforced by the Navitaire code. Dr Chiu says that another common way of utilising an ODBC connection is to use Microsoft Access, and of course TableDefinitions.mdb is an Access database.

220. It was suggested that OpenResStructureMaster.xls and the associated TableDefinitions.mdb were written with a view to supply to Mr Vandertol, the intention being formed at the time Mr Vandertol was first approached. There is nothing to support this suggestion, and if that had been the intention it is surprising that TableDefinitions.mdb, which holds rather more detailed data than does the spreadsheet, was lost and not sent to BulletProof.

221. Between 7 and 11 February 2000, a copy of OpenResStructureMaster.xls was made and sent to BulletProof, where it was checked into Source Safe on 17 February under the name Openrestables.xls. It appears to have been sent in response to a request from Mr Vandertol. Mr Vandertol gave it to Dave Horbury[23] to consider much later, in July. Mr Horbury seems to have asked for TableDefinitions.mdb and Mr Just was asked to send it, but he could not find it and there is no trace of that ever having been sent to BulletProof. It has now been found, and is among the documents in the case. Dr Hunt observes that the table names in the first column of the spreadsheet OpenRestables.xls have a structure characteristic of Microsoft Access, which replaces hyphens in names with underscores and prefixes the database name to the dataset name. This is reasonable, and, having looked at TableDefinitions.mdb I conclude that it is possible that Openrestables.xls was derived directly from it.

[23] I infer this from the email at G3[63].

222. Openrestables.xls was the subject of considerable evidence. Two tables in it appear to cover the same ground. The first page is a long list of the detail datasets and the automatic masters in OpenRes, with a description of what most of them contain, or a '?'. The second page is a shorter list going over the same ground, with the same 'Table Description' column, but with a further column identifying only the index fields, that is, those fields with a corresponding automatic master. As will appear, this is part of the study that needs to be done in order to work out the structure of an OpenRes database. There follows a page on which an attempt (incorrect) has been made to work out a relationship between the tables relying on the index fields alone. The final page is of some importance, since it sets out in some detail the overall operation of the database in four different operations: loading and maintaining flights and fares, selling a seat, flight and passenger following and cancellations. Against the description of the operation of selling a seat is a small table setting out two pairs of columns headed 'Read' and 'Write'. The second pair of columns is headed OpenRes. Mr Rybka did not set this table out. It is on the face of it a comparison in database accesses between OpenRes and some other system, on a line by line basis. The speed of operation of a database depends in part upon the number of reads and writes made to it during a particular transaction, and so the numbers are of interest. There was no attempt at the trial to relate it to the operation either of eRes or (more surprisingly) to the Proof of Concept, which was the database intended to show superiority in throughput to OpenRes. Nor was there any attempt to show that it was accurate in respect of OpenRes. It is accordingly suspicious but mysterious.

The Agents table in eRes

223. On 18 February 2000, Mr Vandertol produced Spec2000.mdb, another Microsoft Access database with extensive information relating to the user interface and the maintenance screens of OpenRes. By March, it contained an 'all users functional report' which was a detailed analysis of the OpenRes user interface. It contains further queries about OpenRes. It does not contain information expressly relating to the databases, and it can be disregarded for this purpose. What it does contain are screenshots of maintenance screens, one of which[24] Mr Carr QC showed had been used by Mr Nuttall in the design of the Agents table, which appears to have been substantially complete by 19 May 2000, if the date on ZZ/259 is reliable, since this shows substantially the whole table. The fact is that the Agents table, the purpose of which is to record the basic information in relation to Agents, has a forerunner in the DestinationPoint database as well. Thus, the idea of setting out the attributes of a particular entity (the agent) with a distinct table was clearly understood by BulletProof (indeed, it is entirely obvious), and the precise fields came from this screen, although they are very similar to DestinationPoint. There are only so many ways that one can describe a person. However Mr Nuttall got hold of it, it seems to me likely that this is how the table was generated: not from any sort of direct access to the OpenRes database but from precisely the sort of operation that Dr Hunt and Dr Chiu were agreed was normal in finding out what data a client had. I was told by Mr Carr that this was the only example of use of a screen shot (apart, of course, for the Flight table in the Proof of Concept which was plainly carried through into eRes) to which I would be referred. There are, I think, others, but with one important exception there is no point in examining them as they do not form the subject of any claim.

[24] G1/277: there are a number of screens showing the same data relating to an agent, and one at G1/276 showing Mr Vandertol's questions.

224. Mr Nuttall joined BulletProof at the beginning of April. As I have indicated, I consider him to be a satisfactory witness. He had considerable experience writing SQL stored procedures, having started in 1991 on Sybase SQL Server, the product that became Microsoft SQL Server. He too had worked at Countrywide Home Loans. He had only once designed an entire database. Again, Navitaire were faced with the difficulty of deciding whether Mr Nuttall had no relevant skills or whether he was sufficiently skilled to write the 200-odd stored procedures present in eRes in November 2000 and design the database tables. Table design and writing stored procedures go hand in hand. They decided to suggest that Mr Nuttall had insufficient relevant experience.

225. Appendix 9 to Dr Hunt's October report sets out the state of the tables of which complaint is made at various dates at which they were checked in to Source Safe. She has highlighted in red the first date on which various fields that she says are equivalent to the fields in the OpenRes table were introduced. These highlightings are incomplete and occasionally inaccurate, and most importantly do not highlight changes that Dr Hunt does not consider relate to fields in the OpenRes INVENTORY dataset. I take the example of the first table shown, Flight. Dr Hunt correctly sets out the content of the table as it was on 19 May 2000, shortly after Mr Nuttall's arrival. The 'Lid' item is, in fact, the same item as 'MaxSeats' in the Proof of Concept table, so this column has been renamed to use a term which I was told was commonly used, but of which Mr Nuttall could be expected to be unaware. The other change is to rename LocalDepDtTm and LocalArrDtTm as LocalDepTm and LocalArrTm, and change their datatype from datetime to int. A new field 'LocalDepDt' (datatype datetime) is inserted, but Dr Hunt does not highlight it since she does not relate it to OpenRes. Subject to this trivial change, the table is the same as that in Proof of Concept. The fields missing from the list in version 2.5 of the Scenario Document are Capacity, Airline Code, and the Service Flag for flight status, and these are added in November 2000. After November 2000, only one field that Dr Hunt says was copied from OpenRes was added. That was in December a year later, when the TailNumber field came in. By November 2000, Mr Nuttall had already written several complex stored procedures depending on the Flight table. I do not accept that Mr Nuttall required any assistance from OpenRes tables to construct the July version of Flight. Its source is the table in EZJetFare.er1 and the Proof of Concept, augmented by the fields identified as necessary in the Scenario document.

The FlightFollowing table in eRes

226. Flight-following is a different matter. It is not part of the reservations system accessed by the reservations API. It is in the system for easyJet's own purposes but it is important, since it tracks the progress of a flight. Mr Nuttall says he created it completely in April or May, and did nothing with it. The stored procedures that accessed it were written by easyJet's own programmers in Luton, and not by Mr Nuttall, who said that he had to change the table slightly from time to time as they 'tweaked' those stored procedures. Navitaire say that FlightFollowing was copied directly from the corresponding OpenRes dataset, FLIGHT-FOLLOWING. In order to explain the allegation, I need to describe rather more about the sequence of events.

227. Mr Nuttall appears to have worked on more than one server, and bundle ZZ contains early tables and procedures that he wrote that could still be recovered from his own personal machine. Their names indicate the machines: BulletProofWKS1 is the workserver and 'BulletProofSQL1-KKBETA2' is a machine running a β (pre-release)

version of SQL Server 2000. This machine contained ERD's produced by SQL Server itself. In these files, Mr Nuttall included creation scripts for more than one table, and so BulletProofWKS1.ejworkspace.TAB is apparently dated 19 May 2000 and contains copies of the (a) Agent, (b) ConnectionsState, (c) ConnectionUserAssoc, (d) ProcedureRedirector, (e) SystemAvailable, (f) wrkTravelerAddress, (g) XMLTemplate, (h) ConnectionFareAssoc, (i) Credit, (j) WrkTraveler, (k) WrkReservation, (l) WrkTravelerItinerary, (m) WrkTravelerPhone, (n) WrkPayment and (o) WrkNotes tables.

228. It is instructive to see how the project changed over time. Two years later, in the ejWorkspace database as of 24 May 2002, one may find copies of tables with the same names as those I have labelled (c), (d), (e), (k), (l) and (n). The Agents table I have discussed above (paragraph 223). The WrkNotes of 2000 is equivalent to WrkComments in 2002, the fields being very similar. The remaining tables are by no means the same, and the way of identifying the traveller and distinguishing the traveller from the contact has changed. But it is idle to suppose that in some way the 2000 workspace (the idea for which must have come to Mr Nuttall very early) is in some way generally a copy of the schema of OpenRes.

229. On 12 July 2000, another file called Openrestables.xls was sent by Mr Just to Mr Kleyman. This was a list of the fields that were being accessed by the easyJet reporting functions, compiled by Amjad Khan, a programmer who worked on easyJet's reporting programs in Luton. The programs EZTODAY and EZREPORT are programmed by easyJet and access the database directly, as was explained by Mr Webster and Mr Just. There was a cross-examination of Mr Webster which I thought was going to support a suggestion that easyJet should not have written EZREPORT. If the suggestion were made, it would be obviously absurd. The data accessed was easyJet's data, about flights by easyJet aircraft. easyJet wanted to know what it was: how otherwise were they to do so? The spreadsheet refers to six tables: INVENTORY, FLIGHT-RELEASE, FLIGHT-FOLLOWING, CITY-TABLE, CURRENCY-TABLE-2, ROOT-DVDN and ROOT-HDVN. The last two have nothing to do with OpenRes, but are part of easyJet's telephone-call handling system. This communication told BulletProof what some of the field names in the tables were. It told them nothing about datatypes, table linkages or anything else. Mr Just explained that the spreadsheets were sent to BulletProof because they had requested the data that are accessed by the programs, as the email itself makes clear. Navitaire's rather odd suggestion is that the tables 'cannot have been used for this purpose' as they contained no data. This suggestion is odd because BulletProof wanted to know what data easyJet accessed. This is part of the design process.

230. Then, on 13 July, an event to which Navitaire attach great significance took place. This was the supply by Mr Just to BulletProof of another MS Access database, bp.mdb. bp.mdb contains four Access tables populated with sample data. The tables are AVAIL-FARES, CITY-TABLE, CURRENCY-TABLE-2 and FLIGHT-FOLLOWING, that is, tables corresponding to four of the five datasets which were accessed by the reporting functions described by Amjad Khan. Very importantly, the database also contained ODBC links to 14 tables in OpenRes including the four from which the sample data is derived. The others are FLIGHT-RELEASE, INVENTORY, PAX-DETAIL, PAX-NAMES, PAYMENTS, PNR-ADDRESS, PNR-DETAIL, SCHEDULE, SEGMENT and SEGMENT-HIST. Oddly, the copyright in none of these other than INVENTORY is alleged to be infringed, and, for reasons of dates, Navitaire are obliged to allege that the defendants copied INVENTORY from another

source to produce their Flight table as I have described above. Among the populated tables neither CITY-TABLE nor CURRENCY-TABLE-2 is said to be infringed either.

231. The evidence was that if these ODBC links were opened, the field names in the corresponding datasets in OpenRes would be displayed, even though the link was itself non-functional. I accept this, although I cannot reproduce the effect with the copy of bp.mdb supplied to me by Navitaire on a portable computer. It forms the cornerstone of Navitaire's attack on Mr Nuttall, which I shall now endeavour to explain.

232. The point is the FlightFollowing table. The hypothesis advanced by Navitaire is that it was written from bp.mdb by Mr Nuttall between 13 July and 19 July, when it was saved to Source Safe. Mr Nuttall's answers to questions about this table are relied on in support of the contention that Mr Nuttall was not to be believed.

233. FlightFollowing does not follow the OpenRes table in content or order. Had it done so, I might have accepted Navitaire's contention. But I have regard to these facts:

i) The orders of the tables are different.

ii) the eRes table has to be seen as containing fields corresponding to items in not one but two OpenRes tables, [redacted].

iii) There is no equivalent in either of the OpenRes tables to the eRes fields ActDepGate, or ActArrGate.

234. The presence of fields whose equivalents are to be found in two OpenRes tables is most significant: how would Mr Nuttall, knowing little or nothing about the airline industry, arrive at this combination working from bp.mdb and why would he include ActDepGate or ActArrGate, which are not present in OpenRes at all? The Navitaire answer is that he must have asked Mr Vandertol, but that does not meet the case. How did Mr Vandertol know? It seems to me that the obvious answer is that Mr Nuttall did not use bp.mdb but was working either from a list provided to him, from a maintenance screen or from a report that combines fields from these two tables: and I find that the FLIGHT-FOLLOWING maintenance programs called FFMAINT and WJFLIFO produce screens that do exactly that. The latter seems to have been of particular interest to BulletProof, since Source Safe contains at least three sets of screen shots of the screens dated in October and November 2000.[25] On the other hand, I have been unable to locate a report that contains these fields, so that possibility can be discounted.

235. The next obvious possible source is Amjad Khan's spreadsheet concerning EZREPORT and EASYTODAY, since most of the accessed fields are listed there, albeit attributed to the FLIGHT-RELEASE dataset, since it appears from the OpenRes schemas (summarised by Dr Chiu most helpfully in his comparative table at Figure 7 in appendix 3 to his third report) that there is some duplication between FLIGHT-

[25] The November screenshots were considered by Dr Hunt: see paragraph 6.112 of her September report. They are printed at G2[47]/512–514. The fields on page 513 that come from FLIGHT-RELEASE are Tail Number, Fuel OUT, Fuel Uplift and Fuel IN. On page 514, the fields are Bag Count Fwd and Bag Count Aft.

FOLLOWING and FLIGHT-RELEASE[26] and DEPT-TIME and ARRV-TIME appear also in the INVENTORY dataset. It is instructive to compare the order of the fields in Amjad Khan's spreadsheet and in the eRes FlightFollowing: they are rather similar, in blocks. On the other hand, there is no mention in the Amjad Khan's spreadsheets of the FIDS-STATUS fields, to which there is an equivalent in FlightFollowing, and (in the context of the movement program) a mention of the STATUS-UPDATE fields, to which there is no equivalent in these tables, but which do find equivalents in the FlightStatus table, of which no complaint is, apparently, made.

236. I accept Mr Nuttall's explanation of the origin of his FlightFollowing table: somebody gave him a list of things to go in it. I find it likely that this list came from something like Amjad Khan's spreadsheets. It is also possible that it came from maintenance screen shots like those exhibited to Dr Hunt's September report in paragraph 6.112 (footnote 25 to this judgment) or may even have been actual screenshots, marked up to show the fields that easyJet wanted. There is no other explanation for the combination of data items from two OpenRes datasets in a single eRes table. This is, of course, a similar process to that followed for the Agents table and is the process described by Dr Hunt and Dr Chiu. I am satisfied that Mr Nuttall did not use bp.mdb in designing his table. I should add that his observation that the contents are dictated by SITA, that is, the data feed from the service provider at the airport, is no doubt correct but is only a part of the story. It is necessary to consider the selection of data items present in eRes, and I have indicated why I think this selection was directed by easyJet.

The use of bp.mdb

237. bp.mdb was immediately sent to Mr Horbury. Mr Horbury was an independent specialist in data migration who was employed by BulletProof for six months from July 2000 to advise on the data migration problems. He had nothing to do with database design. It seems to me, largely from Mr Just's evidence under cross-examination, that Mr Horbury managed to access the OpenRes database and to generate some extract files, by operating remotely on the system at Luton and arranging for the files to be sent to him in California. He analysed the data and then sought more: Mr Vandertol emailed Mr Just on the 27 July:

> Dave is making good progress on analyzing the data. However, without a complete set it is not possible to determine whether a field is always unused or it is just unused in the set of data that we have. Is it possible to get the whole data set prior to your leaving. I want to ensure that the data migration can continue on its merry way.

238. A tape of the entire database was sent in August. It forms the subject of a distinct allegation of copyright infringement, since it is alleged that it was not a backup tape.

239. I find that the significance of the supply of bp.mdb was in the migration of the OpenRes data to the new system. It seems that Mr Vandertol and Mr Kleyman thought that the best way to migrate the data was to analyse what was held in the existing database so that it could be precisely migrated to the new system. It is

[26] The fields apparently duplicated are PASSENGERS, FLIGHT-CREW, NON-REVENUE.

important, though, to bear in mind that Mr Horbury did talk to Mr Nuttall, albeit not much. Mr Nuttall gave these answers:

> Q. You yourself were not personally involved in the data migration other than the occasional assistance to Dave Horbury. A. That also is not quite accurate. I am the SQL expert at BulletProof Technologies and I was the liaison for Dave. I was involved all along in that regard. Dave did not have access to our environment a lot of the time and I would have to get things for him and do things. I believe that all the discussion about the e-mails that I was included on is all about the fact that I believe that my boss realised that this contractor may not survive the length of the project and that at some point, even though I asked not to take it on 100%, it was going to fall in my lap. So he sent me e-mails that included pieces of information that were apparently from his perspective important to that particular aspect of the project. So I received those and did not do anything with them because I knew that at some point I may have to use those.
>
> Q. So is paragraph 39 correct then? From what you have just said, I take it it is, or not? A. We had to define our terms. What do we mean by data migration? Data migration is a huge separate project in and of itself. If easyJet was a start-up, if they had come to us and said we want to start an airline, we would not have to do a data migration. But since we did, the very first task is to figure out how we are going to get the data out of this unknown system, the system that none of us knew anything about, how were we go to get it out, since it was such a large amount of data, how were we going to process it in while they were still changing it and be able to go live and not shut them down for two weeks while we ran this long process. So with the migration task, the first thing was figure out if we can get the data and then start deciding how we were going to get it out. Then once we had some of the information from that system, how do we map that to our environment? How do we map that to our structures? What kind of transfers have to occur? All of that code had to be written and tested and then the process itself had to be run over and over again in order to make sure that we did not have any issues not only with the process but with the actual migration of the data, that we were actually migrating all the data that was required.
>
> Q. That was an exercise in which you were involved? A. That was an exercise over which I supervised and lent my expertise. As Joe Vandertol said in one of his statements, Dave was not a SQL guru. If you look at the code that Dave wrote, it was pretty ugly and most of that was reworked when we did the GO migration. What I helped him do was to try to put it together into a cohesive process.

240. I find that Mr Nuttall's design of the database tables was not influenced by Mr Horbury, and was not influenced by bp.mdb, which was plainly provided for the purposes of the migration. Although Navitaire criticise Mr Just's ex post facto explanation for sending bp.mdb, it seems to me that its subsequent use gives the best indication of its purpose. The one person who plainly used it was Mr Horbury, and I entirely accept Mr Nuttall's explanation.

241. Mr Nuttall said that his design for the database was fixed by June or July. This gave rise to a vast dispute, since Navitaire wished to demonstrate continued influence on Mr Nuttall's design of database from bp.mdb and also from his discussions with Mr Horbury and Mr Wallace, the other person involved with migration.[27] The effect of the cross examination was to show that Mr Nuttall's recollection was not correct other than for (1) the keyserver database, (2) the flight database and (3) part of the workspace database (although the reservations database will be generally the same in structure as the workspace database). There was generally no support for check-in functions by the end of June, but they are added (to the reservations database) in June or July. None of this matters. The other tables of which complaint is made are not copied from the database schemas, with one exception. I shall consider them here.

FlightFare/AVAIL-FARES

242. For the purpose of this analysis, it is necessary to use Dr Hunt's May report on similarities and Bundle F1A tab 11, which is Appendix 11 to Dr Hunt's September report and purports to set out the Table Creation Scripts for the tables in Schedule G. This is what was used during the trial, but I must record that it does not contain the whole of the creation scripts for any of the eRes tables. These appear to be drawn from Appendix 9 of Dr Hunt's report, which purports to set out the 'Final Table Creation Scripts for eRes 12th June 2002'. It contains the note that 'I have removed Go and setuser commands from the scripts to save space.' Unfortunately (Dr Hunt apologised quite frankly for this) the scripts also all omit the 'constraint' clauses that give important information relating to the structure of the database. These should be present in Appendix 11 as well as they help explain certain of the fields. To take an example only, I set out the whole of the relevant part of the script for the FlightFare table:

[redacted]

243. In this code extract, the constraints are set out (cf F1A tab 11 page 295) and named. The constraint clauses provide that the unique key of the table is FlightFareID. It is standard practice, it seems, with BulletProof, to give the primary key the name of the table followed by ID, and such a key has a defined datatype, EntityIdentifier. All entities (rows) are identified by a primary unique key, provided by the so-called Key server. FlightID is constrained to be a foreign key, defined for each row in the Flight table in the ejFlight database. This can been seen from the naming convention I have described and from the fact its datatype is 'EntityIdentifier'. The constraint clause on FlightID means that it has to be one of the FlightID's already defined in the Flight table: this way, the dbms enforces consistency. There may be more than one FlightFare row that refers to a particular FlightID.

244. The corresponding dataset in OpenRes is as follows:

[27] He never spoke to Mr Wallace, but communicated with him by email.

[redacted]

245. There are eleven arrays of 12 items in this dataset, meaning that with the other data items it has 137 members. SQL Server does not support arrays. So the whole table would have to be translated by the copyist. The primary key (if that is the correct term) is on the date of the flight (FLIGHT-DATE). No date, as such, is a key in the eRes table.

246. The evidence on this was given by Ms Antry and Dr Hunt. Both were agreed that the tables occupied the same position in the overall structure of the database, by which they mean that the table associates a fare structure with each flight (i.e. the INVENTORY dataset in OpenRes and the Flight table in eRes). This is done via the multipart key FLIGHT-KEY (which itself contains data) in OpenRes and FlightID in eRes. eRes has further tables for SpecialFare and FareClassCode. From the beginning, Mr Nuttall spent a lot of time working out how the easyJet pricing system worked, as the stored procedures in Bundles ZZ and CC show, and this is reflected in the structure of his tables.

247. Dr Hunt expresses the view that 7 of the 9 columns in FlightFare correspond to columns in AVAIL-FARES. She does not say how in her September report, but in her October report she describes the FlightFares as a 'normalised' version of the OpenRes dataset.

248. It was put to Mr Vandertol that he had obtained the structure of AVAIL-FARES, and so of FlightFare, from Mr Horbury, the fact being recorded in an email of 24 July. But the table had been in existence since mid-May, so that possibility can be excluded. Mr Nuttall had written a complex stored procedure for availability using Flight, FlightFare and two tables called FareValue and FareTranslation by 31 May[28]. easyJet price their fares according to the number of seats already sold, and the price of a seat increases as the number sold. The prices are divided into bands, and the logical problem confronting the programmer is to provide for the correct result when the number of seats sold in a particular booking crosses a band boundary (they are all sold at the higher price) or covers an entire band. The way in which easyJet actually priced their flights had been known to Mr Vandertol since the time of the Proof of Concept. There is no basis, in my judgment, for supposing that the FlightFare table is copied from the OpenRes database.

Queue/QUEUE-RECORD

249. The eRes table and the OpenRes dataset are remarkably different in appearance. I am asked to conclude that the eRes table was copied from OpenRes although all the column names are different and there is a different number of columns. Mr Sundaram, its author, did not give evidence.

250. The table supports a function that enables either uncommitted bookings or bookings which need to be altered to be held in a queue for attention. Dr Hunt expresses the view that the OpenRes dataset was copied 'via the functional specification'. In other

[28] FlightFare_Availability.sp in easyJet/devcode/Databases/ejFlight/StoredProcedures. The two tables are shown in the ERwin entity-relationship diagrams at ZZ tab 57, 59 but not 60, where they have become FareClassCode and FareClassPriceTranslation.

words, Mr Sundaram designed the table to support the function. For reasons I shall enlarge on below, this is not enough: there is here no infringement.

HistoryLog/HISTORY

251. The eRes table is very similar to the OpenRes dataset. This was produced much later in the development process. Mr Nuttall had originally designed a per-booking table history database and the nature of the table can be seen in Dr Hunt's spreadsheet on evolution of the tables. easyJet asked instead for a simple change log and it turned out to be impossible to migrate the existing easyJet history data into Mr Nuttall's proposed structure. He then designed two tables called ChangeLog and ChangeTypeCode. He mapped the fields in the HISTORY database to the new table, and changed the names in his new table. So he copied the dataset, although he ended up with additional columns. The ChangeTypeCode table kept its name. The HistoryLog table is a copy. I draw some comfort from this for the views I have expressed in respect of the other alleged copies: once one identifies what Mr Nuttall used, the nature of his use becomes clear. This is the only table that obviously owes its structure to copying from the OpenRes database.

Topic/TOPIC

252. There is a difference between the systems here. eRes stores all the topic and subtopic information in the database. OpenRes stores the subtopic information in files outside the database, and the subtopic dataset includes a filename, rather than a text entry. I see no reason to suppose these are copied: their essential nature is immediately apparent from spec2000.mdb.

Flight-Key and Rec-loc

253. OpenRes uses two principal keys, FLIGHT-KEY and REC-LOC. They are not pure numbers as are the primary keys in eRes, but contain data. FLIGHT-KEY is a 32-character field of which the first eighteen characters identify a single flight: they are composed of date, city pair and flight number. The next 14 characters identify up to two connecting flights. FLIGHT-KEY-A is an automatic master and acts as an index to AVAIL-FARES, SCHEDULE and FLIGHT-STEP. easyJet used only the first 18 characters, since it did not operate connecting flights, to identify flights. In July 2000, easyJet asked BulletProof to include FLIGHT-KEY in eRes because the reporting functions at easyJet used the field to identify flights. An 18-character column called FlightKey was accordingly included in the Flight table (ejFlight database) and the TravelerItinerary table in ejReservations. It turned out later that the use of the key was essential to maintain data integrity during migration. eRes does not use FlightKey as a key. It is just a column entry but its contents are maintained to be correct.

254. easyJet added a BookingNumber field to the Reservation table. This is the unique identifier for the reservation and is the booking number given to the customer. Every Reservation has a ReservationID provided by the Keyserver, and the BookingNumber was generated from this number by converting it to base 36 (i.e. digits 0-9 plus 26 alphabetical symbols per digit). The vowels are removed to prevent happenstance offensive words, and what remains is prefixed with an E. This was the BookingNumber.

255. When the existing data was migrated, the REC-LOC, which is another key in OpenRes, but is also the booking number given to the client, was stored in the

BookingNumber field. This way historic data at the time the system went live would have bookings which could be identified using the identifier that easyJet had supplied to the user. It is not used as a key.

Structure of the database

256. I can cut this discussion short. The only tables which form the subject matter of a complaint of copying are the eight specified. There was an attempt to broaden this out into a general discussion of tables 'with a medium similarity to OpenRes'—see appendix 11 to Dr Hunt's January 2004 report. In my view, the conclusions of section 6 of the January report are clearly against any suggestion that the Contact, Comment or TravelerItinerary tables owe their existence or their position in the database to any 'knowledge of OpenRes'. Indeed, Dr Hunt quite fairly and correctly traces their origins back to DestinationPoint, even showing that one or two of the items upon which she relies as showing similarities between OpenRes and eRes can be traced back to that system. It also demonstrates that the contents of TravelerItinerary were initially decided on before any access is alleged to the relevant dataset (SEGMENT) in OpenRes. FlightCheckin, FlightDelay and FlightStatus, as I have indicated, were all checked in to Source Safe on the same day as FlightFollowing, and contain fields corresponding to those in FLIGHT-FOLLOWING and FLIGHT-RELEASE. As I have also indicated in the discussion of FlightFollowing, no plausible basis for the division of these fields on the basis of bp.mdb has ever been suggested. These tables were not copied from the database, and I suspect, again, that they are directly related to the screen shots and to Amjad Khan's spreadsheet (paragraph 235 above).

257. It seems to me when it took place the copying was straightforward. It was from screens, or perhaps from lists of data items like Mr Khan's spreadsheet. There was one copy from the database itself. The allegation of copying the database structure owes its origins, I think, to what Ms Antry considered the good idea about OpenRes, which was the idea of associating the price directly with the flight by use of the INVENTORY, AVAIL-FARES and FARE-CLASS-M datasets. This enabled per-flight and per-number of remaining seats pricing, which Ms Antry described as unique, in contrast to the normal industry practice of per-route pricing. I have no doubt that this feature is considered important by Navitaire, but once it is said that prices are to be associated with flights an important element of structure is disclosed.

258. As I have already indicated, the OpenRes schemas do not provide any description of the linkage between the various datasets otherwise than inferentially. Thus, if it is known that REC-LOC is the reservation number (as it is) it will be clear that there is a linkage between tables where REC-LOC is a field. The same goes for FLIGHT-KEY and FLIGHT-SEG. These are fields upon which the relevant tables are indexed, and it is the fact that they appear in more than one table which provides the reader with a clue as to their use. But as the database is not a relational one (albeit that it is provided with an SQL interface that interface is not used in OpenRes), no more information is provided. There is a potential many-to-many relationship between datasets sharing keys. The manner in which the datasets are in fact used is a matter for the code that manipulates the data in the datasets. Dr Chiu explained this in great detail and, I thought, most compellingly. The proposition which Navitaire insisted on advancing was that the entity-relationship diagram produced by Dr Hunt reflected something express in the schemas. It did not. What it reflected was an analysis of the schemas with a knowledge of the underlying business requirements and reference to the source code which was not available to BulletProof and easyJet. Thus, Mr David Evans, who

was really responsible for the design of the OpenRes database said when confronted with Dr Chiu's diagram showing all the possible relationships that he would 'never conceive to write this up this way myself' his attitude was entirely understandable, because he knew how the database was used. Dr Chiu expressed the view that the more-or-less complete absence of manual masters in the OpenRes database meant that it operated as a series of indexed tables with common indexes. Where the same index key appears in more than one table, he took the view that it represented a *potential* many-to-many relationship through the automatic master corresponding to that key, and he showed this in his physical data model diagram. He relies upon Ms Antry's statement, the cross-examination on which I quote, describing the linkage in OpenRes:

> Q. Fair enough. Let me pass on from that. Can I ask you to look at paragraph 5.11, over the page. Picking it up at the second sentence, you say here, "This 'linkage' is typically done through complex rules and coding has been designed to fulfil specific business requirements. Frequently this is achieved by the logic of the application which provides the links and dependencies between the datasets with their value and true meaning". When you refer to the logic of the application, you are referring there to the COBOL code? A. Yes, the processing. How the transaction flows through, yes.
>
> Q. You give a helpful example in paragraph 5.12. I am going to slow down just a touch at this point because one needs to get one's head around what you are saying. You say, "The PAYMENTS dataset has a link to the PAX-NAMES dataset through the PAX-KEY data field. This key field is defined as the record locator with a suffix of '01' meaning the first passenger on the reservation. To someone who does not understand the logic behind this linkage, they will incorrectly assume that each payment is linked to a specific passenger when it is stored into the database, which is the more typical industry standard, where payments are directly linked to specific passengers. Our system does not link PAYMENT records to specific passengers, only to the first passenger on the itinerary, as a matter of tracking for the credit card system". Having understood what you have said there, my question is this. Someone who is just looking at the data would never understand that, would they? A. They could see that by looking at the data in the database.
>
> Q. How are they going to see that? A. They could see that you have a reservation with four passengers and that you paid with two credit cards and both of those credit cards would be linked to PAX 01 on that reservation.
>
> Q. That is if they spot the payment with the two credit cards. A. If they are looking at the data, they are probably going to be looking at anything related to the reservation. So by taking the record locator, they would be able to find the payment records.

> Q. It is a bit of detective exercise, is it not? A. It depends on what they are looking for, yes.
>
> Q. Are you not right when you say that what they are more likely to do is just to assume that each payment is linked to a specific passenger? A. That would be the easiest way, just to assume that.
>
> Q. And if they make that assumption, then they would not spot it. In other words, what it comes down to is do they just assume or are they going to go on the detective exercise? A. If they make the assumption, then anything else based on the assumption would be wrong.

259. This is an example of a relationship that can only be ascertained by looking at the code. David Evans made the point even more clearly when he said that you did not need the automatic masters. You would still have the entity-relationship diagram as Dr Hunt had drawn it, but all the linkages would be established by the applications code.

260. Navitaire submit that Mr Nuttall did not know how to link his tables until he had seen the OpenRes data structure (submission on facts paragraph 164). This suggestion is extravagant. He understood linkage between tables (as did Mr Vandertol): you cannot write a stored procedure without understanding linkage, and the submission overlooks the undoubted fact that it is difficult to extract the information from the OpenRes database. In my judgment, the structure of the database was not copied.

261. At this point, I should refer to the development of the eRes API. This was primarily the work of Brian Milliron. He was a good witness and obviously a highly competent programmer. Mr Milliron studied the OpenRes system by accessing it remotely. He also studied spec2000.mdb. His structure for the API is as follows. It is divided into a number of ‘workflows’. Each workflow represents one of the basic tasks of the system: availability, booking, check-in and so on. The user interfaces (easyJet.com or the VT100 interface module) call the appropriate workflow and pass an XML message. The message is processed, and the appropriate stored procedures to obtain or write the specified data are called using the BPTDataEngine, which contains five standard methods (DBSelect, DBInsert, DBUpdate, DBDelete and Execute) for executing the stored procedures. Mr Milliron had started on the booking and availability workflows soon after he joined, and they were checked into Source Safe in May 2000. The development of the API does not have any direct influence on the database, and, as Mr Milliron says, you can add a column to the table and have no effect on the API. But it shows substantial development going on and, more importantly, programmers with a good knowledge of the processes of the system. It seems to me to be unlikely that against this background there should have been little progress on database design beyond Proof of Concept (now six months old) by mid-June 2000, as Navitaire submit.

262. Finally on this subject I should observe that Navitaire emphasised the opportunity presented to Mr Nuttall to benefit from a study of the OpenRes database to which he was given access during the migration process. I have looked with some care to see when BulletProof actually received a copy of a schema similar to the report relied on by Navitaire as evidence of the copyright work (Document 210). This would, of

course, have been immediately available to easyJet by the use of Adager or Suprtool, to which they had access. Remarkably, I do not think that it was until August that BulletProof received the metadata extracted from the OpenRes database from Mr Wallace in this useful form. Since this is the best guide to the OpenRes database, easyJet's failure to give it to BulletProof is some evidence that copying the database was not at the forefront of everybody's mind.

263. I accept entirely that Mr Nuttall had partial access to the OpenRes database itself through bp.mdb, and as a result of his discussions with Mr Horbury, but the two databases are too distinct to be the result of an analysis of the structure of the OpenRes database and its translation into a relational form with an associated large-scale alteration in field names and datatypes. Dr Chiu was of the view that such a process was more complex than designing the database from scratch using what he considered to be the usual sources, and I accept this. Moreover, many of Mr Nuttall's tables went through many variations, and most are not said to be copies at all. The claimant is unable to detect any copying of a substantial part of the OpenRes schemas in the PersonName, Reservation, ReservationCharges, Payment, CreditAccount, Equipment, FlightCheckinBagTags, AirportCode, FlightCheckin, and TravelerItinerary tables. Indeed, by 1 November 2000, seven months after Mr Nuttall joined, there were 88 tables, big and small, in the system, and 307 stored procedures, long and short. These are largely the work of Mr Nuttall. Not only do these tables support easyJet's business model, but they do not contain those features of OpenRes that were of no interest to easyJet. To copy the OpenRes database while omitting the features of no interest would be a task requiring much effort, and there is no suggestion anywhere that Mr Nuttall approached the design from this direction. Where it is being suggested that the copyist lacked the ability to design the tables on his own, a high degree of similarity to the copyright work, in consequence of a lack of ability to depart with confidence from the pattern, is to be expected. I cannot accept that the similarities in structure between eRes and OpenRes, such as they are, arose from Mr Nuttall's access to the OpenRes database.

264. Generally speaking, however, it seems to me likely that screenshots were extensively used to identify the characterising data in the system, together with questions of the 'What is this for?' or 'Is this required?' kind. The documents show a large number of them. I shall deal with this aspect of the case after I have dealt with the migration of data, since the legal issues are in part common.

Migration of data

265. I can now return to Mr Horbury's activities. As I have indicated, he was supplied with bp.mdb, but complained on 17 July 2000 that it needed an ODBC connection that he did not have. He also asked for TableDefinitions.mdb, but never seems to have obtained it. Instead, Mr Horbury obtained access using Citrix to easyJet in Luton, no doubt thereby being enabled to access their database access software. He was warned that he had access to the live system, and was only to do queries on indexed columns. He did some extractions, which left files on the machine in Luton which were subsequently sent to him.

266. It must have become clear that it was not going to be possible to extract data in this way, and in late July BulletProof engaged Ron Wallace to extract data from the tape to which I have referred (paragraph 238 above). Mr Wallace was asked to extract the

data from the tape and convert it into a format capable of being understood by SQL Server. Mr Wallace's data was cleaned up and used as test data for the eRes database.

267. The process of migration is described in detail by Mr Horbury. A number of intermediate databases were devised for the purpose of data migration, at least one of which was a direct SQL Server implementation of some of the OpenRes datasets, that is, a database which mimicked the manual master and detail datasets in OpenRes. The first tape was unreadable, but a second sent directly by easyJet to Mr Wallace could be read, and he transferred the whole database on to his employer's HP3000. I do not think that there was any discussion of the precise format of this tape, but it appears to have been a complete dump of the database, capable of being restored to an HP3000. It was extracted using Suprtool by easyJet. It would necessarily include all the metadata as well as the data, or it could not be restored. Mr Wallace used Adager to produce a report of all the schemas in the restored database, and on 19 August 2000 he sent it to Mr Vandertol, offering to help with its interpretation. It turned out that Mr Wallace, though a good technician, was not familiar with concepts such as entity relationship diagrams or cardinality, and I am invited to find that his intervention though no doubt well-meant was irrelevant to the task of designing the eRes database. I do not think he could have helped substantially, and I am sure he did not do so.

268. Mr Horbury used the schema files to identify the datasets which contained data that needed to be migrated into the new database. For the purpose of migration, he designed two intermediate databases. The first was an SQL one-to-one equivalent of OpenRes. The second, called Work Area, was a database into which data from the first, manipulated and then transferred to the eRes database. Mr Horbury initially carried out this operation on the extract files produced by Mr Wallace, but he received data extracts direct from Mr Colin Rees at easyJet in Luton and via Mr Vandertol, extracted some data from specified datasets into 'comma delimited files' and sent them to Mr Horbury. Mr Horbury proceeded to create so–called DTS packages to load the data from the bulk files obtained by Mr Wallace (which contained the OpenRes field names) into SQL Server. He ultimately succeeded in doing this. When he had worked out how to migrate this data into eRes tables, it was used as test data. This work provided BulletProof with the schema for an intermediate database (the Work Area) and the necessary packages to load the OpenRes data from the comma-delimited files into an SQL database.

269. Only one technical point arises here. Since the tapes include the metadata of the OpenRes database, that metadata can be extracted from the database once it has been restored from the tape into an HP3000 machine. This was done, and the result emailed to BulletProof in August 2000.

270. Eventually, Mr Horbury identified a comparatively limited number of OpenRes datasets containing data that needed to be migrated, although he produced in all about 60 DTS packages. Eventually, I think it is accepted that 18 OpenRes datasets were loaded into the OpenRes SQL database, from which 14 were migrated to WorkArea and thence to eRes. They did not include any of the fixed data tables, and AVAIL-FARES proved impossible to migrate because Mr Horbury could not understand it: its contents were transferred manually. The migration process was slow, and Mr Rees thought it was not well advanced in September 2000.

271. The system was installed on new servers at Luton in December 2000 (a pre-production version had been first supplied in June). Before this happened, there had

been several trial migrations, and many had failed. Comparative reports were written and run to show the degree to which migration from the OpenRes system was successful, and KPMG ultimately carried out an audit of migrated data: this turned into the condition precedent for the eventual go-live of the system. Much to easyJet's increasing impatience, migration efforts continued through 2001 and finally the system went live on 14 December 2001, about 10 months late.

The Law

272. For this purpose, it is necessary to consider sections 50B and 50D of the 1988 Act. It was contended that the tape sent to Mr Wallace was a back-up tape, but this was not shown. It may have been in back-up format, but it was not shown that it was made for back-up purposes, and it was not used for back-up purposes. So section 50A does not assist easyJet.

Decompilation

50B.—(1) it is not an infringement of copyright for a lawful user of a copy of a computer program expressed in a low level language—

(a) to convert it into a version expressed in a higher level language, or

(b) incidentally, in the course of so converting te program, to copy it,

(that is, to 'decompile' it) provided that the conditions in subsection (2) are met.

(2) The conditions are that—

(a) it is necessary to decompile the program to obtain the information necessary to create an independent program which can be operated with the program decompiled or with another program ('the permitted objective'); and

(b) the information so obtained is not used for any purpose other than the permitted objective.

(3) In particular, the conditions in subsection (2) are not met if the lawful user—

(a) has readily available to him the information necessary to achieve the permitted objective;

(b) does not confine the decompiling to such acts as are necessary to achieve the permitted objective;

(c) supplies the information obtained by the decompiling to any person to whom it is not necessary

> to supply it in order to achieve the permitted objective; or
>
> (d) uses the information to create a program which is substantially similar in its expression to the program decompiled or to do any act restricted by copyright.
>
> (4) Where an act is permitted under this section, it is irrelevant whether or not there exists any term or condition in an agreement which purports to prohibit or restrict the act (such terms being, by virtue of section 296A, void).
>
> **Acts permitted in relation to databases**
>
> **50D.—**(1) It is not an infringement of copyright in a database for a person who has a right to use the database or any part of the database, (whether under a licence to do any of the acts restricted by the copyright in the database or otherwise) to do, in the exercise of that right, anything which is necessary for the purpose of access to and use of the contents of the database or of that part of the database.
>
> (2) Where an act which would otherwise infringe copyright in a database is permitted under this section, it is irrelevant whether or not there exists any term or condition in any agreement which purports to prohibit or restrict the act (such terms being, by virtue of section 296B, void).

273. There are a number of problems. The first is to identify the copyright work. ‘Database’ for the purpose of Part 1 of the Act, is defined in section 3A(1)

> **3A.**—(1) In this part ‘database’ means a collection of independent works, data or other materials which—
>
> > (a) are arranged in a systematic or methodical way, and
> >
> > (b) are individually accessible by electronic or other means.
>
> (2) For the purposes of this Part a literary work consisting of a database is original if, and only if, by reason of the selection or arrangement of the contents of the database the database constitutes the author’s own intellectual creation.

274. So far as the OpenRes database is concerned, the ‘database copyright’ subsisting by virtue of section 3A does not, in my view, extend to the schemas or material directly entered at the Adager or Suprtool prompt which has the effect of changing the structure of the database by adding or subtracting fields, or adding or removing datasets. Such scripts, schemas and even directly entered commands seem to me to be properly viewed as computer programs, because that is what they are. The effect of

Adager and similar tools is to compile or interpret the commands given to it to create data (the database metadata) that will be recognised by the database management system as specifying the existence of items which need to be stored and indexed in a defined way. It is of course true that the schema is not a procedural but a non-procedural language, but that does not seem to me to affect the issue. The metadata defining the fields and datasets, or the tables, rows and columns, falls with some difficulty into section 3A, if only because they are not a collection of data. I cannot help but feel that section 3A is directed to the contents of the database. The one pointer against this conclusion is to be found in the European Parliament and Council Directive (96/9/EC) of 11 March 1996 on the Legal Protection of Databases ('the Database Directive') which section 3A is intended to implement. Recital 15 says 'Whereas the criteria whether a database should be protected by copyright should be defined to the fact (*sic*: the French text is "devront se limiter au fait que", which is clearer) that the selection or the arrangement of the contents of the database is the author's own intellectual creation; whereas such protection should cover the structure of the database.' In an electronic database, there is no compelling need to view the programs or scripts creating the database as part of the database, even though they define its 'arrangement' and 'structure'. Anyway, they acquire copyright even if no database is ever generated from them, and my inclination would be to say that they do so by virtue of the fact that they are computer programs.

275. Whether or not the foregoing is a correct analysis does not seem to me to matter, since the primary obligations placed on easyJet are those of clause 4 of the agreement (paragraph 19 above) and in particular clauses 4a, 4c and 4d. Thus, to the extent that easyJet are entitled to abstract the material they sent to BulletProof, any clause restricting their right to do so is void. I repeat those clauses here for convenience:

> 4. **Conditions of License.** This license is granted subject to the following conditions:
>
> > a. Title to and ownership of *OpenRes* shall remain with Open Skies. (easyJet, however, shall own and retain title to all underlying easyJet data generated by *OpenRes*). easyJet acknowledges that *OpenRes* is the property of Open Skies and that easyJet's rights in and to *OpenRes*, or any portion thereof, may not be assigned, licensed, sub-licensed or transferred (whether by operation of law or otherwise) without the prior written consent of Open Skies, which consent shall not be unreasonably withheld. This restriction on transfer shall not apply to (a) a merger in which easyJet is the surviving entity or (b) a change in the ownership or control of easyJet.
> >
> > …
> >
> > c. Any *OpenRes* software that is supplied to easyJet in machine-readable form may be copied, in whole or in part, by easyJet only for back-up or archive purposes. No other form of copying is allowed.

> d. easyJet agrees not to disclose or otherwise make available *OpenRes* or any portion thereof, or any related material or information, to any person or entity outside of easyJet without the prior written consent of Open Skies. The granting or withholding of such consent shall be entirely within the discretion of Open Skies; however, such consent shall not be unreasonably withheld by Open Skies. Open Skies agrees that it shall not disclose any commercially sensitive information relating to easyJet without easyJet's prior written consent.

276. The abstractions of data of which complaint is made took place in this jurisdiction, in Luton. I say nothing about what happened in California, since that cannot be an infringement of the UK copyright. However, the crucial event, which is the final migration, took place in this jurisidiction. easyJet should be in no better and no worse position than they would have been had BulletProof been located in Skegness rather than Pasadena, but, of course, BulletProof are not liable for acts done in California which would be infringements if done here.

277. The primary question is one of necessity. I construe this word so as to exclude the merely desirable or convenient, but not so as to require the merely absurd—cf. *Sony Computer Entertainment v Paul Owen* [2002] EMLR 34 (page 742) (Jacob J). So far as anything was extracted and supplied to BulletProof, if it was necessary for the purposes of access to or use of the database to extract the field and dataset names either alone or along with the data, then that is enough. Navitaire say it certainly was not necessary. For the purpose of migrating data, Ms Antry says that a neutral file format could be used. For this purpose, the designer of the new database designs the database, having found out what he can about the contents of the old database from legitimate sources. He then tells the user what data to extract into the new database, and provides a file format of unnamed records separated by separators, such as commas. The user extracts the data into this file without telling the designer of the new database what he is extracting. The data is then transferred from this 'neutral' file into the new database. There is no doubt that this method is used—indeed, it was accepted by Mr Rees that its use was possible.

278. It should also be noted that because easyJet did not seek Navitaire's consent, their only escape from a finding of infringement in relation to the data migration databases is either that no substantial part was taken or that what they have done is a permitted act. It does not matter that Navitaire would have had no grounds reasonably to refuse consent.

Screen shots

279. I consider the screenshots at this point. While it is possible that to copy a screenshot cannot communicate anything about the structure of the underlying database, it may incidentally do so and did so in the present case, for example in relation to the AGENTS dataset. Thus, it is possible to allege that the schema for the dataset was indirectly copied by copying the names of the data items shown on the screen.

280. It seems to me that the screen shots tell the user what data was present in the system, but not how it was stored. I think the use of that data is legitimate. It is no more than telling the user what data of his is stored in the system, and he can scarcely access the contents if he does not know what is there. So the use of the screen shots is necessary to find out what data is present, and accordingly permitted by section 50D even if the result of using the screens for this purpose is the reproduction of a dataset schema, or part of it. I am reinforced in this view by the fact that Dr Hunt and Dr Chiu, with long experience in this area, and the textbook, all regarded the use of screen displays as a normal way to find out what data was stored in the system. Indeed, I think any other conclusion would be intolerable. The database would be transformed into a heap of numbers and letters to which only the owner of the copyright held the key.

The copy tables FareClass.tbl and Fares.tbl

281. These are the two tables in the Proof of Concept folder obviously derived from the OpenRes tables. They should not have been extracted by easyJet, if they were, and they should not have been supplied, if they were. Since they were manifestly never used, they are of no concern.

Mr Amjad Khan's spreadsheet

282. This identifies not only the nature of the data but where it is stored. It does so because this is data used by the external routines, and Mr Khan included the dataset name as well as the nature of the data item, which appears pretty clearly from its name. The names will have to be changed to accommodate the new database structure but he was entitled to tell BulletProof what the data items were. I do not think it was necessary to specify the names of the datasets, and so this is a breach of contract, albeit trivial. I do not think it was an infringement since a substantial part of the work was not taken.

bp.mdb

283. It is of course acceptable for easyJet to extract their own data, and identify what it is to BulletProof. If they use Access to do it, they get Access versions of the OpenRes datasets. I think that this is just ordinary abstraction of data. To provide the ODBC links confuses matters, since they are an invitation to BulletProof to abstract data duly authorised by easyJet, but, of course, they include information about the database fields even when there is no access to the data. It is difficult to shoehorn this into the statutory provisions in an acceptable way, but I conclude that to send the ODBC links was not acceptable, and was an infringement.

The tapes and the OpenRes SQL database

284. There is no doubt that a neutral file format technique would have worked. Dr Chiu and Mr Rees of easyJet agreed. It would not have involved sending the tapes to BulletProof. It would still have been open to easyJet to specify further fields that they wished to see in the eRes database, once they had seen a draft of it, to hold data they wished to have or preserve. There would equally have been no objection to BulletProof's indicating to easyJet the order in which it wished to see the data on the neutral format tape and explain what that data was. I suspect that using this technique, and identifying the data to be migrated at an earlier stage, would have taken no longer than the approach ultimately taken, although the witnesses were adamant that the corruption of the data made migration difficult, and that the structure of some of the datasets could not be understood. This technique involved infringement, but in my

judgment since it is a clear corollary to this finding that some other technique would have worked and within a time that was not unreasonable, it caused the claimant no loss.

285. Finally in this connection I should refer to a contention advanced by Ms Antry, to the effect that ‘screen scraping’ would have enabled the defendants to abstract the contents of the database. I have to say that I regard this suggestion as bizarre and unreasonable. ‘Screen scraping’ is a technique in which the output of, for example, a report is not sent to the screen but instead to a file. It would take forever. Dr Hunt was far from enthusiastic for it. In principle, it is no doubt possible, but the test for necessity in section 50D would be met were this the only alternative to what easyJet actually did.

CONCLUSIONS

286. Except in limited respects concerning the process of migrating the data to the eRes system, and the supply of certain database extracts to BulletProof, this action fails.

Annexe 1 Code for the OpenRes availability command

287. The code fragment that identifies the arguments to an availability command seems to be the following in the source module AVAILSCR:

[redacted]

288. I have emboldened certain passages relevant to the simple A13JUNLTNAMS↵ enquiry. We enter the routine knowing what the operator typed is stored in a location called `ans-buff`, and the individual characters, counting from the left hand end, can be referred to as `ans-buff(1)`, `ans-buff(2)` and so on. We know already that the first letter (`ans-buff(1)`) is ‘A’. We need to know whether the command is the AA, AS or AR variant, and so `ans-buff(2)` is checked to see: lines 067700 and 068300.

289. If both the second and third characters are numeric, it is likely to be a date. On the assumption that it is, the program accordingly assumes that the fourth, fifth and sixth characters are a three-character month (JUN) and a routine is called (`0999-convert-date`) that expresses the day (which is in `ans-buff(2)` and `ans-buff(3)`), the month and the year as an 8 digit number (lines 069200-069500). It is assumed that the next six characters are a city pair, and they are stored as such. This is what Mr David Evans described in the passage quoted above in the judgment at paragraph 30.

290. The code fragment also shows the parsing of the contents of `ans-buff` to see if the optional characters ‘/’ or ‘+’ are present, and if so sets a flag for return flights on appropriate dates (lines 071900 ff.).

291. The code to process the availability command in eRes is found principally in the ProcessAvailability.bas source code module and in the AvailabilityObject.cls class module. It is more complex, but it is possible to see that the characters of the command (which in eRes start with the number of passengers immediately following the A, as in A0113JUNLTNAMS↵) are processed on assumptions as to the structure of the string of characters. In other words, the syntax of the command is implicit in the parser: an error in the syntax of the command should produce an appropriate error message from the parser.

292. I am satisfied that the only similarity between the two systems in this respect is in the results of the processing, and in the command lines to be processed.

Annexe 2 Parsing in OpenRes

293. I set out the ‘top level’ of the parser from the file `BUILDPNR` with comments to indicate what is happening.

[redacted]

294. The purpose of setting out the piece of code above is to show how the programmer approached the parsing of the command lines. It is a step-by-step approach: start with the first letter. If the first letter is ambiguous, in that other families of commands, or individual commands, also begin with that letter, advance as far along the line as is necessary unambiguously to identify what the command is, and then branch to the code that processes that family of commands. It is in the latter code, as I have discussed above and in Annexe 1 below, that the syntax of the command is implicit.

Annexe 3 Parsing in eRes

295. As I understand it, the parsing system in eRes works like this. The commands are divided up into single letter, two-letter, three-letter and four-letter commands. They are all placed in arrays, by code like this (from the file `EzVT100CmdParser.cls`):

[redacted]

296. The code which does this (I give a single example, the three-character command names) methodically sets them all out:

[redacted]

297. The array `CCFourCharCmdArray()` contains all the four-character commands, and so on.

298. In order to parse a command line, the method used is first to employ a longest match technique: the parser starts by trying to find a match with the four-character commands, then the three-character commands, then the two-character commands and finally the single character commands, stopping as soon as a match is found. This works because all commands (including sub-commands) differ in at least one letter. In this process, all dot commands are recognised only as dot commands, and generally all commands consisting of a prefix and a variable suffix are dealt with after the invariant part of the command is recognised. Thus, at the first stage, A and A- are recognised as distinct commands, one of one character, the other of two characters. A*n*A, A*n*S and A*n*R are recognised as an A command and they are differentiated at a subsequent parsing stage. First I give the relevant part of the parser: it is a succession of calls looking for a match, longest first. It is part of a function called `ParseAndExecuteCommand.`

[redacted]

299. I next give an example of the function which does the comparison: it compares the first letter of the command string with each of the single-letter commands stored in the array: if it finds a hit, it calls the `ParseCC2char` routine to parse the whole command string, placing the result in a variable called `strResponse.`

[redacted]

300. If the command is A*n*A, it is recognised by all the code above as an 'A' command, and it is when the `ParseCC1char` routine is called that A*n*A, A*n*R and A*n*S are ultimately distinguished. If the command is 'A' all that `ParseCC1char` does is call a routine called `ProcessAvailabilityCmd.` This is very long and is contained in its own source code file, `ProcessAvailability.bas.` It is this routine that checks that the number of passengers has been entered, and then processes the rest of the command string. The A, the S and the R do appear in the code (more than once), and upon analysis they can be recognised as the so-far-unprocessed characters in the subcommands, but they are never seen as a unit with the initial 'A'.

Annexe 4 The eRes code for the incomplete reservation screen '.A'

301. This is the output of a '.A' command. It does not affect the booking: it provides the operator with a list of the things that are needed to complete a reservation. The example of the use of eRes that forms Dr Hunt's walkthrough (the slideshow) demonstrates its use after (1) an availability command A (2) a G1/1 command[29] to grab one seat on flight on line 1 of the results of the availability display command (3) an NT1 command[30] to set the number of travellers. At this point, the operator wants to know what else to do. When '.A' is typed, the things that still need to be done are displayed, with the appropriate codes. There are three parts of the display: the things that need to be done part, the segment (flight) part and the comments part. The code that formats the upper part of the 'incomplete reservation' screen in eRes is apparently contained in the function `ProcessDisplayCmd` in the module `ProcessDisplay.bas`. This contains all the '`Needs`…' lines. Each is formatted to include the spaces that mean that the commands ('`N–`' and so on) appear in a neat column on the screen.

302. The lower part of the screen seems to be generated by the function `FormatGrabSeats`, in the `ProcessGetSeat.bas` module. Lurking in here is code which sets the order in which the items on the line appear (it is quite easy to see the '`0`*`n`*`-EZ`' at the beginning of the line). The line is built on the fly: the order in which the data items are obtained determines the order in which they are displayed.

303. Again, therefore, the eRes code does not contain any sections in respect of these commands that look like the picture that appears on the screen. If one reads the code, one can see how the machine will execute it so as to produce a particular appearance. It is possible to recognise parts of the layout. Paragraphs 8.14 ff. of Dr Chiu's report show this.

[29] Not a permissible OpenRes command. The / must be replaced by a 'class code', in fact 'l'.
[30] Not present in OpenRes

Annexe 5 Conclusions on the display aspects of OpenRes and eRes

304. I am obliged to consider each of the displays in turn. Numbers 1–15 are supplied marked up in pink, green and yellow, at Appendix 6 to Dr Hunt's September Report, which is in turn a photocopy of section 5.2 of her May report. The colour coding is explained in paragraph 9.5 of the September report:

Green: Identical fixed data on both eRes and OpenRes screens

Pink: Similar fixed data on both the eRes and openRes screens

Yellow: Variable data which is set out in the same order and

positioning in both the eRes and openRes screens

305. There are errors in certain of the markups (see paragraph 10 of the January report) but they do not seem to me to matter all that much. Of rather greater importance is that Dr Hunt occasionally suggests that features are identical in the fixed data, when they are not.

306. Numbers 16–21 are GUI user input forms, and are dealt with separately. They are described in section 5.3 of the May report, marked up in Appendix 6 of the September report at pages 164.31 ff.[31]. We revert (numbers 22–27) to the VT100 display. Originally dealt with by Dr Hunt as part of the 'Business Logic' of OpenRes and eRes, these screens are found in section 4.

307. It is common ground that the basic constraint on the design of a VT100 display is that the screen has 24 rows of 80 characters. So if a line is not to be wrapped, or further screen-fulls of data displayed, there must be 24 or fewer lines containing 80 or fewer characters including the displayed data. Dr Chiu described how this can be done when programming on COBOL, by using a coding sheet, but this method does not appear to have been used by the persons responsible for devising the displays.

308. With that introduction, I can turn to the individual displays starting with the VT100 screens.

i) **Item 1. '.A' screen (incomplete reservation):** May report paragraph 5.1. This is a display function, the elements to be displayed depending upon the commands that have previously been executed. It is shown in section 5.1 of Dr Hunt's 2 May 2003 report.

a) Ms Beesley says it took her a day to devise, and I must accept this since she wasn't challenged on it. There is no proper evidence of any prior material in written form for this command, although it is fairly likely that Ms Beesley prepared a list of prompts in an appropriate order. Thereafter, it seems to have been down to the programmers—see 1 Beesley paragraphs 286-291.

b) It includes a further feature to assist the operator. If the operator types just one of the specified commands (say 'N-') without any arguments,

[31] Internal page numbering. Bundle F1A tab 6 pp 202ff.

the system returns an example line setting the necessary arguments out (see OpenRes slide 9). eRes does this as well (eRes slide 11), but nobody told me where the code was, or whether that code reproduced the little example string (it does: for 'N-' the code is in `ProcessTraveler.bas.` It is in `BUILDPNR` in OpenRes).[32]

c) The .A command is similar to the *A command in SABRE, a fact of which Ms Beesley was unaware, thinking that the SABRE command is *R. There was no evidence that any other system has the ability to provide the detailed arguments to a command requiring arguments if it is typed without arguments, although that is an entirely normal, one might even observe usual, feature of any command line interface.

d) Dr Hunt gave this display a 75% score in section 9 of her September report. As an indication that the displays are not very similar, that seems to me to be fair. From the point of view of conveying information, they are very similar, but most of the information is dynamic.

ii) **Item 2. The call centre HELP screen:** Section 5.2 of the May Report. Score 80% in September report. This screen merely consists of a list of available common commands and a one or two word description. The lists are not the same, save where eRes possesses the same command as OpenRes, when the entry in the list is identical. I regard this two-column list as trivial. It is contented that the layout involved 'creative skill and labour' in the decision to list the items alphabetically and in two columns rather than as going onto a second page. Dr Hunt identifies the words 'HELP MENU', 'NAME' and 'COMMAND KEY' as identical between the two screens, together with the two column layout, together with the word 'Vers' in the top right-hand corner. I was not shown the code.

iii) **Item 3. The reservation display:** Section 5.3 of the May report. This is the same display as i) above when all the data has been entered. It is another example of the display produced by the '.A' command. It is to be noted since it records nearly all the essential data in respect of a booking. There are differences between the displays, but the overall scheme is very similar. The descriptive words (Booked, Modified, EZ Record Locator, Received, Lang Curr, Dist and so on) are for the most part the same. To my eye at least, the eRes screen is marginally tidier, but there is not much in it. There was no proper evidence that any of the '.A' screens involved the use of preliminary drawings, and this one is the same.

iv) **Item 4. Check-In passenger counts:** May report section 5.4. 100% in September report. These are identical, but trivial. They involved creative skill and labour to a negligible degree.

v) **Item 5. Check-In Passenger list:** May report section 5.5. There is no evidence as to who devised the layout of this list. Mr McDaniel was proposed, but he could not remember. It is trivial.

[32] It may be surprising that this was not relied on as an example of copying the syntax of a command. It is not, as it is not a piece of code that is executed.

vi) **Item 6. List flights for checkin:** May report section 5.6. The columns are

```
15jun02 Flt St    Dept Arrv   ETD ETA     Dept Arrv   Sold
1) LTNAMS 221     0620/0825                           64
```

that is, line number, segment, flight number (omitting the EZ), status, Departure Time, Arrival Time, Estimated Departure Time, Estimated Arrival Time, Actual Departure Time, Actual Arrival Time, and seats sold. Again, this is trivial.

vii) **Item 7. Display Flight Information:** May report section 5.7. The September report gives it a mere 10%, but says 'eRes includes the OpenRes program name "Fnflinfo"', which is another module. I can discern no similarities between the eRes and OpenRes screens, but Dr Hunt says that another OpenRes screen (shown as 5.7(c) and the only one copied in Appendix 6 to the September report) was also copied. I do not agree. The eRes screen is plainly different from both those mentioned by Dr Hunt. The second screen, produced by a module called `FNFLINFO`, is not explained, save that `FNFLINFO` is a distinct executable program whose purpose is to produce this flight information and does not seem to be executed in response to any of the commands we have been considering. I believe it is executed from item 9 on the top-level menu of both systems (see slide 1 of the OpenRes walkthrough and slide 2 of the eRes walkthrough), but there is no evidence. Dr Hunt's cross-examination on this display was informative:

> MR. ARNOLD: Could you move on please to page 162. This is item 7, display flight information. A. Yes.
>
> Q. We see the respective screens at 163 for eRes and 164 for OpenRes. A. 164 and 165 because ----
>
> Q. Indeed. A. They are not desperately similar as you will no doubt be telling me.
>
> Q. They are pretty different, are they not? A. Yes, apart from the peculiar "fnflinfo" bit [at] the top.
>
> Q. Why did you include these two therefore? A. There is a similarity there. As I have said already, this is not an analysis of how similar something is. There is a similarity. It is included.

In eRes, the screen is produced in response to the FI command. The screen alleged to have been copied is not so produced in OpenRes. I do not know, and was not told, how the code in eRes that places the word fnflinfo on the screen works, where it is, or why it matters. I regard Dr Hunt's response in this passage as completely unhelpful. It suggests that it is not possible to attach importance to her schedule of similarities: anything, however insubstantial, may have been included. Every suggestion must be analysed. Anything like this should not have been included, or should have been discarded as soon as possible, and certainly not in the grudging way exemplified in this cross-examination.

viii) **Item 8. Contact details:** Section 5.8 of the May report. The September report gives it 70%. The cross-examination of David Evans revealed a substantial amount of programming effort underlying the display, but accepted that the display itself was 'like addressing an envelope.' Unsurprisingly the eRes display is similar, but, perhaps surprisingly, not identical. As a design of screen, I think that this is trivial, but of course the programming effort is not necessarily trivial at all. It may be noted that the fields of the screen are entirely natural, and reflect the corresponding contact database table in eRes.

ix) **Item 9. General Information systems:** May report section 5.9. Dr Hunt gives it 100%. The screen is displayed in response to the 'GS' command, and contains a list of general information topics (they include such things as firearms, wheelchairs and company news). Ms Antry gives evidence that it was sketched out on paper, but I do not know if that was for layout (it is two columns) or merely for order. The number of entries in a column, for example, is constrained by the fact that there are 24 lines on a VT100 terminal. Dr Hunt's 100% is inexplicable. The two tables are not the same. The entries are in many cases, I believe a majority, different. All that can be said to have been taken here is the two column layout and the manner in which the system works, i.e. by inviting the user to type a number for the specific topic they wish to read. It should be noted, however, that the eRes screen contains a reference at the top to 'gssearch 1.5'. This is a reference to an OpenRes program which is not used, or copied, in the eRes system. For those who wish to believe that they are still looking at OpenRes, it certainly adds verisimilitude. It appears to have no other function.

x) **Item 10. Display comments:** May report section 5.10. The similarity identified by Dr Hunt consists only of the use of the word 'Comments' and that the comments are listed (in a slightly different format) below that word. This, again, is trivial.

xi) **Item 11. Display bag counts:** May report section 5.11. These are the displays:

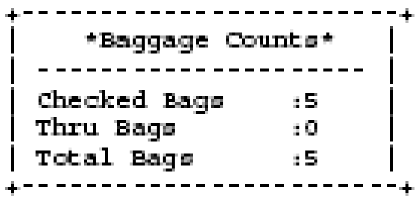

```
+-----------------------+
|    *Baggage Counts*   |
| --------------------- |
| Checked Bags     :5   |
| Thru Bags        :0   |
| Total Bags       :5   |
+-----------------------+
```

Figure 7: eRes bag count display

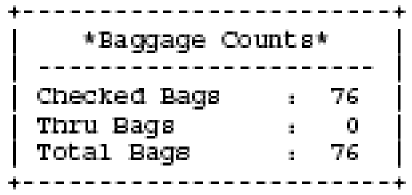

```
+------------------------+
|    *Baggage Counts*    |
| ---------------------- |
| Checked Bags     :  76 |
| Thru Bags        :   0 |
| Total Bags       :  76 |
+------------------------+
```

Figure 8: OpenRes bag count display

a) There is no evidence as to who devised this display. The best evidence is that of Greg McDaniel, who said that he devised much of the checkin functionality—but this does not relate to functionality, rather, it relates to displays. Oddly, it is one of the displays whose layout appears plainly in the code (module `CICLNT01`):

[redacted]

This code is called directly in response to the '.B' command in Check-In, and the module claims to have been written by Mr McDaniel. So, applying the usual presumption, he is its author.

b) The eRes copy with which I have been supplied marks this display as not having been implemented in March 2002, but the display is in fact generated by `DisplayBagSummaryReport` in the module `ProcessAirportCommands.bas`. This is as follows:

[redacted]

c) [redacted] A single string containing all of it called strResponse is assembled, from the response in XML to a message to the application server. It is completed by the prompt for a new command. Another part of the code displays it and enters a state where a new command is expected.

d) It should be noted that the XML attributes representing the numbers are called CheckedinBags, ThruBags and TotalBags. The values of these attributes are ultimately computed by a stored procedure (FlightCheckIn_BagsReport.sp) from a single table in the database called 'FlightCheckInBagTags' by counting all the bagtags issued for the specified flight and converting the result into XML. Since easyJet did not permit through bookings, the value of ThruBags is always zero, and thus TotalBags (the sum) is always the same as CheckedinBags. I draw attention to this only because it is by far the simplest example of a database query in eRes that I have seen.

xii) **Item 12. Display History:** May report section 5.12. Note that the marked up copy of the displays in Annexe 6 to the September report is incorrect, the prompts before and after the display having been coloured green. What the displays have in common are the words 'history', 'Date', 'Agent' and 'Received'. The lines are similarly numbered, and the 'history codes' are the same. The databases in eRes and OpenRes store extensive information relating to the changes made to particular reservations and so forth, and this forms a self-contained part of Navitaire's complaint. I need only record here that the command is '.H' and it displays the history of the current reservation, on numbered lines. The '.H*n*' command, where *n* is the number of a line on the history display, displays the detailed history of that line. The the history details (Booking-Agent, Received-from, Change-date, Change-time, Code and a repeat of the history item) are the same in both systems: eRes has an additional line for the agency-number of the agent making the change.

xiii) **Item 13. Detailed history:** see xii) above.

xiv) **Item 14. Display calendar:** May report section 5.14. Dr Hunt gives it a score of 40% for similarity. There is here no evidence of copying: the layout is conventional, and different (one calendar starts its weeks on a Sunday, the other on a Monday). It is just a calendar. The layout is trivial, no reasonable inference of copying of the layout can be drawn, and it should not have been relied on, except to demonstrate the overall similarities of the user interface.

xv) **Item 15. Show queues:** May report section 5.15. The 'queues' in question are ordered lists of items requiring attention. They have a name and a description, among other attributes. The evidence is that the use of queues is common in reservation systems. There is no fixed data in the screen display: Dr Hunt gives it 100% similarity because of the padding with spaces and the use of colons. As Dr Hunt acknowledged in cross-examination, even on this basis the displays are not identical (eRes lacks the second column of colons). They do display similar data. So far as what is displayed is concerned, the OpenRes display is the QUEUE-ID (6 characters), a colon, the QUEUE-DESC (26 characters), two spaces and a colon, followed by a count. This is a direct display of the contents of two fields of the OpenRes datatset called QUEUE-TABLE. The eRes display is, it appears, derived from the QueueType table in the workspace database, the corresponding field being identified by Dr Hunt as the QueueTypeCodeName field. This has 30 characters, and the display appears to be truncated to 29 characters. So there is that difference as well.

xvi) **Items 16–21:** these are graphical interfaces from the Schedule planning module. They are dealt with separately below, since in their corresponding displays BulletProof have copied certain icons devised by Navitaire.

xvii) **Item 22. Airport Menu.** May report, section 4 page 4.56, Display A. The eRes screen is at page 4.15 Display A. The marked-up copies are September report Appendix 6 page 164.44 (OpenRes) and 164.43 (eRes). This gets 90% from Dr Hunt. It is a list of functions available to Airport staff, and is in the familiar two-column layout. All but three of the 16 functions available in OpenRes are available in eRes, and the absence of those functions represents the difference between the two screens. The screen was, in fact, specified by easyJet to Navitaire (OpenSkies) from a list of what could be included. As Mr Farrukh Khan acknowledged in cross-examination, order and content came from easyJet:

> Q. You explain in your first witness statement about how the menu was customised to easyJet's requirements using an existing program. Is the existing program that you used for that purpose the MENUX reference? A. That is correct.
>
> Q. So you would have asked easyJet what options they wanted and in what order and they would have told you what they wanted and you then went away and used the program to produce this. A. Right. (page 1102).

I conclude that if anybody is the author of this layout so far as its contents are concerned, it is whoever at easyJet gave Mr Khan his instructions. The two-column layout, the numbers and the 'Enter choice:' prompt are trivial.

xviii) **Item 23. Flights Display.** May report 4.57 Display D (OpenRes), 4.16 Display B (eRes), September report appendix 6 pages 164.45 and 164.46. There is no fixed data in these displays at all. They are the response to the 'A' command in OpenRes and the 'A*n*' in eRes (see above paragraph 26.ii)). The output of this command differs between the systems only in the final columns, where OpenRes allows for more than one class: eRes, in contrast, does not. The displayed data is, in order, line number, airline code, date, day of the week, number of available seats, city pair, departure time, arrival time, number of stops, price and seats available. It seems that eRes always displays a 9 in the first seats available column. No detectable skill and labour goes into this display that I can discern.

xix) **Item 24. Flights list.** May report Section 4, page 4.60, Display N (OpenRes), Section 4, page 4.19, Display J (eRes), September report appendix 6 pages 164.48 (OpenRes), 164.47 (eRes). This is the output from the Flight Close Report program (item 4 on the front OpenRes screen) and is best seen from the slideshow (slides 42-46 in OpenRes, slides 40-42 of eRes). This is just a list of flights. The only fixed data are the words 'dep', 'arr', 'avail seats' and 'sold'. The display is arranged in columns, and seems merely to list the identifying features of a flight. There is nothing in this. Moreover, to get to the display, eRes and OpenRes are rather different, both in the prompts and what is typed at the prompt. In eRes, the display is not generated by a different program.

xx) **Item 25. Flight Manifest.** May report Section 4, page 4.61, Display O (OpenRes), Section 4, page 4.20, Display K (eRes): September report, Appendix 6, page 164.50 (OpenRes), 164.49 (eRes). This is a report generated in OpenRes by entering a line number in the prompt following the last screen (Item 23, Flights list) in the Flight Close Out program. There is no evidence as to how the eRes display is generated, and I cannot find the code. Overall, obviously the displays are very similar.

xxi) **Item 26. Display Changed Reservation.** May report: Section 4, page 4.75, Display H (OpenRes); Section 4, page 4.32, Display F (eRes). September report Appendix 6 page 164.52 (OpenRes), page 164.51 (eRes). These are substantially identical with the Reservations display (item 3, paragraph 308.iii) above) and do not call for separate comment.

xxii) **Item 27. Display Reqested Flights.** May report, Section 4, page 4.86, Display D (OpenRes), Section 4, page 4.42, Display D (eRes): September report pages 164.54 (OpenRes) and 164.53 (eRes). This is the display following an irregular operation. There is fixed data, although that in eRes is not referred to in Dr Hunt's September report. It is display C on page 4.41 of the May report. It is different from that of OpenRes, although it identifies a similar operation. The data display just identifies the passengers, and their identifiers.

309. The GUI 'screens' (items 16-21) are rather different. There is no difference of principle from the VT100 screens, but they are more obviously graphic works in their own right, and they have been rather more closely copied. While the screens are fairly simple, in OpenRes they represent an interface to a program (the Schedule planning program) rather different in concept from the remainder of the system. It is a client module for use by the database administrators at headquarters for changing the persistent data in the database relating to the times and dates of flights and seat prices.

It interacts with a server module (FMSERVER) written in COBOL. The evidence was that the program was the result of merging a fares maintenance module with a schedule maintenace module.[33] This was not the first GUI that had been devised at Navitaire but the evidence is that some time was spent on the appearance of the screens because the intention was that this appearance would be consistent across the system when Navitaire produced a GUI interface to replace the VT100 screens. The applications including the screens were built using Microsoft Visual Studio, and the resulting application is in the C++ language. As one might expect, this application permits the designer to 'prototype' the screens, that is, construct screens that will display dummy data and accept input, before the underlying software is written. It permits the designer to design icons for the buttons, and some specific icons were designed by Justin Wilde. These icons were copied by BulletProof. The screens of which complaint is made are the 'front screen' and a number of the individual screens.

310. I was told little about the eRes module. The evidence is that it was written in Visual Basic. Mr Wheeler gave evidence that there was much of the OpenRes module that did not relate to eRes. The general similarity of the screens is no doubt in part due to the identity in the features of appearance of User Interface components common to Microsoft products. There was evidence from Mr Salib that it was an unpopular application to work on and that it was not much used in practice, easyJet preferring to modify the relevant database tables by importing Excel spreadsheets, a facility that had been provided for them.

311. It is the front screen (**Item 16**: May report page 5.51 (OpenRes) and page 5.50 (eRes)) that contains the icons, which are individually printed out in Schedule F to the Particulars of Claim. The OpenRes screen apparently opens the way to a wide functionality not present in eRes, but, to the extent that the icons relate to functions present in eRes (I do not believe I am told what they are) they have been taken. Mr Haddock's evidence was unchallenged:

> 7. In practice, designing a user interface is a complex task in which there are a substantial number of decisions to be made about how all the various commands and functions will be presented to the user. The general requirement is for the most common commands that a user is likely to use to be put in easily accessible locations on the screen. A good design will not bury frequently used commands several menus deep so that they are hard to find. To do this well is a highly skilled job.
>
> 8. I designed the interface based on screen shots of, and end-user documentation for, the text-based Fare Maintenance. I also used printed definition of data structures from the back-end program to determine all the fields of data I needed to display and collect. I did not base the design of the screen layouts on any third party systems. The Microsoft Visual Studio development environment that I used to develop Fare Maintenance has a very robust design interface for Windows applications. I worked directly in that environment to create prototypes for the various screens presented to the user. These

[33] In FlightSpeed, the new system, Navitaire have separated the functions again.

prototypes eventually evolved into the actual product after discussions with Thomas Lerman and Mary Beesley without any documentation describing the process.

312. There is a number of kinds of skill and labour here: There is the overall design, in which the structure of the various menus is worked out. No work relied on in this case, apart from the code itself, can be said even to record that effort. Then there is the appearance. This naturally is to be viewed as a graphic work, and the draftsman uses the development environment to move the various elements of the interface around on the screen until they are in a suitable position. If he wants buttons with icons, he can either use the ones supplied (the first two on the screen are standard Microsoft icons) or he draws his own. Mr Wilde in fact drew the icons. I have been supplied with the original bitmap and icon (.ico) files. Eight icons are taken from about 18 devised by Mr Wilde. I am sure that enough skill and labour went into the little icon drawings to entitle them to copyright, and they have been copied exactly by the defendants. Their general positioning on the screen in two toolbars is entirely conventional, as anyone who has used a Microsoft GUI will know. However, the menu bar (File/Edit/View/Communications/Options/Flights/Fares/Help) has to be designed, except for the four standard entries. The four buttons (Send All Items/Send Item/ Edit Item/Delete Item) are positioned where they are (rather than below the window, perhaps) as a matter of taste[34]. I think that the whole is entitled to a copyright.

313. The eRes screen is much simpler: the toolbars with the icons are shorter, but the four buttons are in the same place. On the whole I think this is an infringement of the OpenRes copyright.

314. **Item 17**, the Build Non-Stop screen, and **Item 18**, Change Schedule, are very simple screens, but parts of them have been copied precisely. Such skill and labour as there is in the original screens has been taken, and there is infringement.

315. Dr Hunt's May report refers to a further screen, Build Fare Class[35], that is so different between the two systems that I cannot see how it was relied on as an indication of similarity. There is no reproduction of the OpenRes screen. This screen is not relied on for copyright infringement, but is still relied on as one of the similarities between the systems.

316. The remaining screens, **Items 19, 20 and 21**, are in the same case as items 17 and 18. The amount of skill and labour going into them as graphic works is modest, but what has been put in has, equally, been taken.

317. I have not really considered the source of these screens: but it seems likely that OpenRes Schedule Planning.doc, which is the Navitaire document describing the module, and which was checked into Source Safe on 17 February 2000 by Mr Kleyman is likely.

[34] Always assuming that Visual Studio, which enforces Microsoft standards for GUI's, permits it, but there was no evidence on this.
[35] Pages 5.60 and 5.59.

Annexe 6 Reports

Introduction

318. There are fourteen reports complained of out of 66 in OpenRes version 5.58. Certain of them, in Dr Hunt's view are 'key'. There is no challenge to copying, but the results are in no case precisely identical. Again, there is no suggestion that the underlying code is copied, and thus the result depends in part upon the correct characterisation of the work (literary or artistic) and the substantiality of what the defendants have taken. Note that we can here only be concerned with the fixed data as displayed on the screen, and the position of the variable data.

319. In general, the defendants' attack on this aspect of the case is that Navitaire are not responsible for the contents of the reports. In nearly every case, it is said that the contents of the report were specified by others (WestJet, Freedom Air and easyJet itself, in a number of cases). Thus, the selection of what is specified originates with others and the effort went into the COBOL code, leaving little taken by the defendants that could amount to a substantial part of the copyright work, that is, the code.

320. The claimant says (relying on a passage in the cross-examination of David Evans) that to the extent that the contents of the report were specified by WestJet in the person of Don Bell, all that he did was to specify problems. The passage will not support the contention. Taking the reports in order:

i) Flight Close Out (File A0025ROS, choice 4 on the Airport menu). This was one of a number of reports (including Flight Close, Passenger List, Flight Loads, and Payment Totals) specified by Don Bell. David Evans said that 'Don was responsible for a lot of the design of the management information reports that later became part of the Open Skies product line' and this was confirmed by Farrukh Khan. Examination of the code shows how tiresome placing data at the right point on the screen is if COBOL is the programming language. That has nothing to do with the skill and labour going into the data to be displayed, or the column headers (which describe what Mr Bell specified). There are three screens: the first specifies the flight(s), the second is an availability display and the third is the report for the line number specified from the second screen. There is nothing in the first two screens.

ii) Flight Loads (File A0020ROS). Mr McDaniel says he wrote this report, as does Farrukh Khan. The code itself says it was written by Mr Khan, but some of the changes are the work of Mr McDaniel. It is impossible to choose between them: but if all Mr Bell specified was a report showing the number of passengers, that is all this is.

iii) Payment receipt (file LOCPAYRP: Airport menu choice 12). This does appear to be a report designed by Navitaire. It was written before easyJet acquired its system, and, although the file records comparatively minor changes made at the request of other customers, the changes do not appear to be substantial. The report sets out payments per passenger per location, and totals, and totals per location. The fixed data is trivial, the programs report the data in different orders down the page, and the OpenRes code contains a bug which constantly announces it is at the end of a screen when it is not. The similarity lies in what is reported, and in the fixed data across columns, and there is much programming effort in extracting it from the database. Again, the formatting must be distinguished from the code used to extract the information from the database, which is completely different in the two systems. I do not regard this as a substantial part of the code.

iv) Gender Count Report (File FLTRPT01: Airport menu choice 3). The report is remarkably simple, and was designed for Freedom Air. As the code says, it will list the number of male, female and child pax on a given flight. I am afraid I consider that the form of the display involved trivial skill and labour although I accept that the COBOL coding might have been time-consuming.

v) Flight Manifest (file A0022ROS: Airport menu choice 5). The evidence is that parts, at least, were specified by Freedom Air. I am not quite sure it goes as far as Mr Arnold suggests, and was all specified by Freedom Air, but the contents are straightforward, and the result is a simple list arranged into columns. I cannot view this as a substantial part of the code.

vi) Passenger List (file PAXLIST: Airport Menu choice 6). This was programmed by Mr Khan. It was specified by WestJet, and the data to be extracted must, therefore, be disregarded. What remains is trivial. Indeed, if it was a cut-down version of Flight Manifest, as Mr Khan suspects, its substantiality is slight in any event.

vii) Check in (file A0021ROS: Airport menu choice 7). The selection criteria are not the same, since OpenRes asks if the user wants to see the passenger names. The OpenRes screen is 120 characters wide, and eRes 80. It appears from Mr Khan's evidence to have Mr Bell involved in its history, and the contents are entirely banal.

viii) Inventory report (file A009ROS: management utilities). This report lists the seats sold in the form of rows having the format date/flight number/city pair/departure time/arrival time/seats sold in a scrolling list. It is entirely trivial as a piece of layout. The selection of data is obvious. For such a simple report, the COBOL programming effort is substantial, but none of that is taken. The corresponding stored procedure in eRes is only about seventy lines long, but, of course, that program handles the formatting separately.

ix) Fees Detail (file A0035ROS: Management Utilities). The eRes report is formatted using different typefaces (as is the Inventory report above). It offers an additional selection option (summary/detailed) to that of OpenRes. The displays are very similar. Again, this report appears to have originated with WestJet, and the column titles follow the content. The COBOL is again complex.

x) Sales Report (slsbypay: Management utilities). This report was specified by easyJet. Again the programming effort seems to have been substantial.

xi) PNR Balance (A003ROS: Management utilities). This report was specified by easyJet. Again, once the contents are specified, the programmer (Ms Antry) set out the columns and labelled them. The programming effort was substantial, but that is not appropriated by the defendants.

xii) Commission Report (A0004ROS: Management utilities). The contents of the report were specified by easyJet. It is not suggested that the layout was specified, but the layout follows from the data specified.

xiii) Segment Report (EJSEGRTE: Management utilities). Again, the contents of this report were specified by easyJet. The layout is trivial, and follows from the contents specified.

xiv) Payment Totals (A0031ROS: Management utilities). This report is not included in Dr Hunt's 'similarities' report. It is introduced in her September report. I do not consider there to be much similarity between the eRes and OpenRes reports, and the two are straightforward.

321. The claim in respect of reports specified by easyJet was always going to be difficult to justify. While I might have expressed my views globally, I should make it clear that I have considered all the reports, and I find this claim cannot be substantiated. I am impressed by the amount of programming effort that went into the display of very simple layouts on the screen, and in manipulating the data, but the defendants manipulate the data in a different way and display in a different way. The actual layouts are trivial. What the defendants have taken cannot amount to a substantial part either of the individual modules identified above or of the code as a whole.