FUJ00081969

Privileged and confidential

FOUNDATION REPORT

Alan Bates and others v Post Office Limited

Dr Robert Worden and Chris Emery

Definitive version 31 May 2018 Table of Contents

1. EXECUTIVE SUMMARY

1 This Foundation Report is not written for court purposes, although parts of it are expected to be reused in expert reports for the court. Its purpose is to set out for Womble Bond Dickinson, for Counsel, and for the Post Office some key lines of argument which we intend to use in our expert reports, in approaching the central issue of the Horizon trial, which we here summarise as: "Were there bugs in Horizon which could have affected the accuracy of branch accounts?".

2 We expect many of these lines of argument to be developed in our reports for the court. A key objective of our expert reports will be to make these ideas, which we here refer to as 'Foundations', clear and understandable to the Court. Much of the material in this report is intended to be used in our expert reports for that purpose. However, to describe these arguments clearly, in this report we need to address several areas which we understand may be out of scope for the Horizon trial in March 2019. These areas include IT systems which are out of scope for the trial, business processes as they were defined and as they were operated in practice, and facts about the in-service history of Horizon over several years. The ways in which those areas will be addressed in our expert reports for the Horizon trial or addressed in subsequent trials when they may be in scope, can be discussed based on the understanding of this report.

3 This executive summary gives an overview of the core argument which, we currently believe, will enable PO to demonstrate to the court that the level of bugs which could have affected the accuracy of branch accounts was very low or zero; and to show that specific bugs which will be highlighted by the claimants' expert could not have had that effect. The overview of that argument follows next.

4 Horizon is the name given to a set of financial applications run by PO to manage branch business and finances. We are not concerned in this report with the distinction between 'Horizon' as defined for the purposes of the March 2019 trial, and the larger set of IT systems of which Horizon is a part. We need to consider the larger set to understand the many checks between and across IT systems, which would rapidly reveal the existence of certain software bugs, forcing them to be promptly corrected - and in many cases ensuring that they would not have had some harmful effects, even before they were corrected.

5 The purpose of financial applications is to calculate sums of money. It is a universal requirement for such applications, including accounting applications, that they should do so with very high precision - because any inaccuracy or discrepancy, even of a single penny, can be a matter of concern to managers and others. Managers need to have a very high degree of trust in their financial systems.

6 For the Post Office, there are two additional reasons for requiring a very high degree of trust and accuracy
in its financial data.

7 First, the volume of money which passes through PO branches during a year is of the order of £100 billion - which is comparable to the whole budget of the NHS, or approximately 10% of the UK Gross National Product. However, a high proportion of the money which flows through PO branches is 'agency'

work such as bill paying, which yields very small margins to PO. These small margins reinforce the need to track all flows of money very precisely.

8 Second, when there are shortfalls in branch accounts, they are typically borne personally by subpostmasters. To retain the goodwill and commitment of these people, the Post Office needs to be confident that they are not forced to pay for any shortfalls, real or apparent, which were not caused by them.

9 Therefore, both subpostmasters in the branches, and managers and others in PO, require a high degree of accuracy in the figures in Horizon, and a high degree of trust in those figures.

10 The number of customer transactions carried out in PO branches is of the order of 6 million transactions per day. Because many of these involve handling of cash and stock, and manual data entry of the amounts of cash and stock involved, inevitably there will be a certain proportion of errors of data entry or errors in the manual processes. Therefore, the number of erroneously entered transactions is expected to be several thousand per day. It is a key requirement on Horizon that these erroneous transactions should not lead to cumulative errors, or to a progressive 'drift' of the figures recorded on Horizon away from reality. For this reason, there are many error detection and correction measures built into Horizon and the business processes around it. These error correction measures will figure prominently in this report. It is not possible to address the core issue of the Horizon trial without understanding them in some depth.

11 To understand the error correction measures built in to Horizon, it is important to know that the actual financial state of the Post Office and its branches is not defined by the data in Horizon. It is defined by external reality and external sources of data, including:

* Physical cash and stock in the branches
* The Post Office's bank accounts
* Financial transactions (carried out at PO branches) recorded in the IT systems of PO client organisations.

12 The purpose of Horizon is to track that financial position as accurately as possible, based on input from branches and other sources, and from time to time to correct any errors arising from errors in branch input, to give an accurate financial picture to PO management and subpostmasters.

13 There are several strategies for ensuring that errors or failures in the Horizon hardware and software do not affect the accuracy of its financial data. These strategies will be called 'Intrinsic Error Prevention' and they include:

* Numerous checks built into the supporting system software such as DBMS or communication software
* Resilient hardware design
* Testing

- Automated double entry accounting checks (such as zero-sum baskets, and trial balances)
- Keeping a secure audit trail, which can be compared with other Horizon data if anomalies arise
- Many numerical cross-checks, of sums of money or stock calculated in diverse ways, from redundantly stored data in different systems and databases (e.g. a data warehouse) for many different management purposes
- If an error is detected in software or reference data, Fujitsu's and PO's processes for correcting these errors

14 The main strategies for detecting and correcting user errors in branches, which will be called 'User Error Correction' are:

- Measures built in to the user interface to prevent user errors
- Daily and other checks of cash and stock made by subpostmasters; processes for branch balancing and rollover
- Checks of the same things by PO audit teams
- Regular automated comparisons (reconciliations) of the sequences of transactions and their monetary sums recorded on Horizon, with the sequences of the same transactions and sums as recorded automatically by PO's clients, on their own IT systems
- PO managers and staff inspecting any anomalies which are detected by these reconciliation checks, or by other cross-checks described above, and deciding how to reflect them, if necessary, in PO accounts and branch accounts
- This may lead to transaction corrections in the branches.

15 The usual effect of these checks, in the case of any manual error made in a branch, is the following:

- There is a transient error in the branch accounts, arising from the manual error.
- Following some reconciliation check by the subpostmaster, or PO audit, the error is corrected, typically through a transaction correction.
- After the correction, the accounts of the branch are accurate; therefore, the effects of manual errors in branches are only transient.

16 Much of this report will be devoted to describing the strategies for Intrinsic Error Prevention, and User Error Correction. Our expert report will include evidence on how they have worked in practice.

17 To fully analyse the effectiveness of these strategies (i.e. the robustness of Horizon), it is necessary to consider various business process and operational issues, and the reliability of various clients' financial systems, and Horizon's track record in service. Some of these are outside the scope of the Horizon trial.

18 It is an important property of Horizon that the measures built in for User Error Correction also have the effect of exposing, and possibly correcting until they are fixed, large classes of possible errors in Horizon. These include coding errors, errors in reference data, and errors in batch schedules. In this report we will describe why that is, and our expert report will also illustrate it by examples.

19 The consequence of these measures is that for any bug in Horizon (in the code, reference data, or batch schedules) to lead to a non-transient error in some branch accounts and to do so for branches over an extended period, it is necessary to show that the error:

- Has an effect on branch accounts
- Is not rapidly detected by the many checks in Horizon for Intrinsic Error Prevention
- Is not rapidly detected by the many checks in Horizon for User Error Correction

20 These three criteria form a 'triple filter' on any bug - so that any alleged bug in Horizon will need to pass all three tests, if it is to lend any support to the claimants' case.

21 In our view, the nature of this triple filter is the core issue for the Horizon trial. It is essential for the court to understand clearly the filter and how it operates, so that the court can understand the arguments of both parties about any specific alleged bug or incident. Understanding the triple filter is the best basis to protect the court from being mystified by expert talk.

22 In our preliminary opinion, the expected number of bugs which will pass the triple filter is zero or very close to zero, despite the scale and complexity of Horizon. One challenge for our expert reports will be to explain that point convincingly. We may not be able to prove the point completely; but we may alter the balance of probabilities to the extent that a burden of proof is more fully placed on the claimants.

23 For any bug which the claimants' expert asserts affected branch accounts, we will test it by the triple filter. We hope to eliminate all candidates in this way.

24 Malicious alteration of branch data by Fujitsu or PO employees, or by external parties, needs separate treatment, considering the authentication, security, and audit measures built into Horizon; and possibly considering Horizon's service history.

25 Similarly, interruptions of service (e.g. caused by hardware failures) may need separate treatment, based on the resilience measures to handle such situations built into Horizon, its supporting system software, and hardware.

2. INTRODUCTION

2.1 Purpose of the Report

26 This foundation report is not intended for the court. It has the following purposes:

a) We would like it to be read by selected members of the legal team, PO and Fujitsu, to enable them to assess our current understanding of Horizon, and to provide feedback to us on the topics we describe below, and on any other topics as they see fit.

b) Substantial portions of this report will be incorporated in our expert report - to educate the court about IT and Horizon, to help ensure the court is not baffled by obscure technical arguments or jargon in the claimants' expert report, and to ensure that the court turns to our report, rather than the claimants', when it needs to understand Horizon.

c) Another reason for using parts of this report in our expert report is to serve as the foundation for detailed discussions of specific bugs in Horizon, as alleged by the claimants, to show, where possible, that those bugs did not have adverse effects on branch accounts.

27 We expect that the material in this report may be substantially rearranged before it goes into our expert report. For instance, parts of it may be moved to appendices.

28 Readers may, of course, provide any feedback they wish about this report. The feedback we specifically request is as follows:

* From the legal team - whether or how this report may have altered their understanding of the Horizon trial; what parts of the report need to be re-expressed or will be unclear to the judge; and how it may be rearranged in the expert report to fulfil purposes (b) and (c) above. We also ask for feedback about the likely amount of factual evidence we may be able to adduce in support of specific points.

* From the PO - whether the picture we paint of PO's business operations, and the impact of Horizon at the branches, is a realistic one.

* From Fujitsu - to correct any misunderstandings about Horizon, to point us towards documents which may help to improve our material, and to point out any other aspects of Horizon or the ways in which they have developed and support it, which may help the court form an accurate impression and reach the correct decision.

29 Following these reviews, we do not currently intend to issue a revised version of this report, but to incorporate the feedback into our expert report. It may be appropriate to hold one or more meetings with readers of the report, to fully understand their feedback.

2.2 Contents of the Report

30 Section 1 of this report is the Executive Summary.

31 This section 2 describes how the content of the report is divided into sections.

32 Section 3 describes the principles of accounting systems, and how modern accounting systems are implemented as IT systems.

33 Section 4 describes the internal checks which are built into accounting systems, and the external checks that can be made of the data in accounting systems. It describes the measures for Intrinsic Error Prevention and User Error Correction, as defined in the Executive Summary.

34 Section 5 describes the range of business applications which Horizon is required to support, both in the branches and in PO back office processes.

35 The Horizon system has undergone frequent changes, in a complex history since its inception in 1999. We will describe its architecture in two main time periods.

36 Section 6 describes the Horizon system as it was in the period 2000 - 2009. After giving a central 'snapshot' of this period, we then start to describe the most important changes during that period

37 Section 7 describes 'Horizon New Generation' (HNG-X, and later HNG-A), introduced in 2010, in which a major element of the architecture was changed. In HNG, instead of holding persistent transaction data in each branch, all transaction data was held centrally in a single branch database. HNG involved a complete refresh of the hardware and software in the branches. Many central elements of Horizon persisted over both periods, as will be described in sections 6 and 7.

38 Sections 6 and 7 both describe complex Horizon architectures, with many major components (most of which are in scope for the Horizon trial), and interactions between the components which can be partially understood from architecture diagrams, showing how data are passed between the systems. However, this still leaves many topics incompletely described, which will be important background for the Horizon trial.

39 Section 8 builds on the previous sections to address these topics across the whole sequence of Horizon architectures. The topics described in section 8, with reference to specific Horizon architectures, are:

* How the Horizon architecture supports User Error Detection - including stock checks made in the branches, branch balancing and rollover processes, reconciliation with external systems, several kinds of audit, and other management checks which are made on Horizon data.

* How the Horizon architecture supports Intrinsic Error Prevention - including double entry checks and cross-checks of different versions of the same data.

* The uses of reference data and data-driven software

* Architecture and business processes for Reconciliation, Transaction Corrections and Transaction Acknowledgements

* Measures for security and user authentication

- Hardware and software resilience of the whole Horizon system - including hardware redundancy, and resilience of underlying software components such as DBMS and communication software. This is needed to understand the possible impact on branch accounts of various kinds of interruption of the Horizon service, including communication failures.

- Processes for development and testing of Horizon

- Fujitsu and PO processes for supporting Horizon in service, including error detection and correction.

40 Having read so far, the reader should have sufficient information to understand the operation of the 'triple filter', which we will use to analyse the impact of many kinds of error (for instance, in software, or reference data, or in batch schedules) on branch accounts. Section 9 will illustrate the triple filter, using it to analyse one error in Horizon which has been cited by the claimants in their pleadings.

41 Appendix A contains a glossary of terms used in this report.

3. ACCOUNTING SYSTEMS - FUNCTIONALITY, BUSINESS USES, AND ARCHITECTURE

42 Computerised accounting systems have been in use since the 1950s, and accounting is one of the most mature applications of computers in business. The techniques for building these systems are very mature, as are the safeguards for ensuring that they work correctly. This section is an introductory survey of the practice of building computerised accounting systems, and covers:

* The business requirements that computerised accounting systems must meet

* The different users of accounting systems, and the ways in which they use them

* The checks and safeguards built into accounting systems, in their current architectures

* The levels of trust which business users commonly place in their accounting systems

* The reasons why business users place the level of trust they do in their accounting systems

3.1 Business Requirements for Accounting Systems

43 Before computerised accounting systems existed, the profitability and financial health of any business enterprise was tracked by a process of manual bookkeeping. In this process, clerks would manually record every financial transaction of the business in books of accounts (or ledgers) which could be inspected to assess the financial health of the business, or to assist in making management decisions. There would be periodic checks of the information in the ledgers, to check two things:

a) that the ledgers record a self-consistent (and therefore possible) state of the business

b) that the state of the business, as recorded in the ledgers, was in full agreement with some external reality (such as physical stock, or bank accounts)

44 Both checks involved arithmetic sums - either of money, or physical assets, or of commitments to pay money or transfer assets. Each check was a check that two different sums, made from the ledgers or as sums of some external quantities, gave the same amount.

45 If either of these checks failed, there would have to be a process of drill-down, of the following form: "sum A is not equal to sum B. We can examine the components of sum A, and the components of sum B, and place them in correspondence with one another, to find out where the discrepancy arises". Having found the source of the discrepancy (and if necessary having taken corrective action in the business) some correction would be inserted in the ledgers, so that after the correction the ledgers again held a consistent and accurate picture of the business - which again would pass the checks (a) and (b).

46 The requirement (b) - that the ledgers should always agree with the external physical reality - has always existed. It has always been the case that the records written in ledgers are a representation of external reality, rather than reality itself. Therefore, changing a record in a ledger does not change external reality. Changing the ledgers does not alter the true financial health of the business, which depends only on external reality, such as its cash and bank accounts and other assets. However, the state of the ledgers

should accurately reflect that external reality - and if it does not do so, to the extent that it does not, the ledgers are less useful. The ledgers are intended to track reality as precisely as possible, and to require correction (to match reality) as infrequently as possible - because that is a time-consuming and expensive process, which reduces the confidence of those parties, internal to the business and external to it, who rely on the ledgers to understand the state of the business.

47 The requirement (a) - that the ledgers should present a self-consistent picture of the state of the business - has not always existed. It is possible to keep a set of ledgers as one or more lists of assets and liabilities, with each asset or liability recorded only once, so that no internal check of self-consistency is possible[1]... That changed around the fourteenth century, with the invention of double entry bookkeeping

48 We can understand double entry bookkeeping by starting with the simple case of a trader on a farmer's market, who has some money, and who has some sheep. He can keep a list of his money, and a separate list of his sheep - complete with their names if he wants to. He can track changes to those lists, with entries like '23rd July: sold one sheep - Dolly'. From the changes he can work out his current position, in money or in sheep: 'now I have 17 sheep left'. This is single-entry accounting.

49 Double-entry bookkeeping starts when his list of sheep contains two types of information - the sheep he has, and the price he paid for them; and he also keeps a separate list of his money. He tracks the changes to these lists in a series of dated transactions: '25th July: bought one sheep for 5 shillings'. With that transaction, the sum of his money list goes down by 5 shillings, and the money total of his sheep list goes up by 5 shillings. So, he makes two entries - in those two lists - with money value -5 shillings and +5 shillings, and the total money value of the two lists does not change. Because of the self-consistency of mathematics, however he chooses to do those two sums, the sum of the two sums should not change from day to day. This is his double entry 'trial balance'. If the number does change from any day to the next, he knows he has made a mistake - and he can start to track it down.

50 The basic double entry principle is easily extended to more complex cases - when he sells a sheep for more than he paid for it, and so makes a profit; when he borrows some money and incurs a debt, when he owes some tax, and so on. The basic principle remains. Whenever he makes entries in his books for any type of transaction, however complex, he always makes at least two entries in different columns of figures - and does it in such a way that the net value of all the money entries is zero. Then the sum over all the columns of figures should not change from day to day. That is his self-consistency check on the figures.

51 We need to understand why double entry bookkeeping was such a powerful advance, which swept across Europe within a few years of its invention.

52 Because any accounting system is intended to track external reality, and to give the most accurate possible picture of that reality, it is essential from time to time to check the picture of reality, held in the accounting system, against reality itself - the physical assets of the business, its money in cash or banks, its obligations and debts.

53 However, checking against external reality is (or was) an expensive process. You cannot simply look at the

[1] This was the case, for instance, in the clay tablets of accounts found at Ur and Knossos.

books - you have to get up from your desk, go out into the warehouse and count stock, check your bank balance and count your cash, consult other people, and so on. Because checking against reality was an expensive process, it did not get done very frequently. If the check is only made occasionally, and mistakes are found, the interval of time in which one or more mistakes might have occurred is a long one. The error is more likely to have arisen from multiple mistakes. Looking for several mistakes together is much harder than looking for one mistake; there is no 'telltale number' to look for. The process of 'drilling down' to find the origin of a mistake is difficult and unguided, with few clues.

54 Double entry bookkeeping changed this. If a bookkeeping mistake is made, that mistake will lead to a discrepancy against external reality, which will eventually be found in the external reality check. But any mistake is most likely to have occurred in one column of figures, without any balancing mistakes on the other columns. So, the mistake will immediately destroy the trial balance. Checking the trial balance is much easier and cheaper than checking against external reality. It can all be done by sitting down at a desk with the books and an abacus or calculator - so it can be done much more frequently. When a discrepancy is found, it is now much easier to drill down and find its cause. For instance, if each entry in the books is dated (as it will be), by just inspecting the books you can find the exact date and nature of the transaction which was not recorded as zero-sum, and which destroyed the balance.

55 So double entry bookkeeping immediately reduces the cost of keeping accurate and trustworthy accounts. It was an early, and highly effective, form of error repellency - in a time when manual errors of bookkeeping were likely to be frequent. The error repellency guarded against a single point of failure (i.e. a mistake in a single column of figures) by making any such mistake rapidly and obviously visible, in the next trial balance.

56 The use of double entry bookkeeping had other important commercial advantages, as well as reducing the costs of keeping accurate books:

* **Working in a network of trade**: From the beginnings of trade, trade consisted of a network of traders, exchanging goods, services and money between them. For any two individuals or parties to trade as part of the network, they have to agree a basis of trading between them (e.g. in a contract) and they need both to monitor that their trading conforms to that agreement (e.g. that I am charging you the price we agreed, and for that price I have delivered to you the goods we agreed). Trade depends on the two parties agreeing what actually occurs between them, down to a very detailed level (which may be down to the last penny). To check this agreement, day by day or month by month, they each use their sets of accounts. Party A looks in his accounts to say: 'on Thursday I delivered to you 5 widgets' while party B looks in his accounts to say, 'On Friday 5 widgets arrived'. Without this agreement they cannot trade.

Therefore, any set of accounts is repeatedly being checked against several other parties' sets of accounts. Some selected extract from A's accounts is compared with a selected extract from B's accounts, and they should match - both item by item and in monetary sums. Any discrepancies between the two are like sand in the bearings of a machine - they hinder trade, reduce trust, and lead to additional costs. In these circumstances, having the error repellency and increased reliability of double-entry bookkeeping reduces the risk of discrepancies, gives you the means to find the origin of any discrepancy, and gives you a commercial advantage over any competitor or business partner who does not have equally accurate accounts- because you are more reliable and easy to work with.

* **Reduction of fraud and theft**: Any large business is likely to delegate the process of bookkeeping to employees, rather than the owner. These employees see large sums of money passing through the books they keep and may be subject to the temptation to 'skim a little off' for themselves - possibly in small amounts, which they hope will not be noticed. With single entry bookkeeping, this could be a straightforward process. Take a bit of cash and, at the same time, alter the cash ledger so it matches - so that counting the physical cash, and matching it against the books, will show no discrepancy. But with double entry bookkeeping, it is not so simple. If you just alter the cash ledger on its own, without some balancing entry in another ledger, you destroy the trial balance - which will soon be discovered. If you try to be more clever - taking 5 shillings from the cash ledger and adding 5 shillings (the price of one sheep) to the sheep register, that will not be discovered until later, when the sheep are counted - but when it is, the owner may be able to drill down or recall events, to find the exact day on which the discrepancy arose, and who wrote it in the ledger. Alternatively, the owner himself may be tempted to falsify the accounts. Double entry bookkeeping makes this much harder to do, dramatically increasing traceability.

* **Accounting to external parties**: With the growth of capitalism, the typical business was not simply the property of one owner, beholden to nobody else. The accounts were no longer just a tool for that person to manage his own business but were also an essential tool to explain the state of the business to external stakeholders - such as shareholders, banks who lent it money, or governments who taxed it. To say to these people: "you can look at my accounts or audit them, to check the truth of what I tell you about the business", the self-consistency check of double entry accounting became an essential tool - a pre-condition for checking and trust.

3.2 Discretionary Elements in Accounts

57 A primary use of an accounting system is to enable external stakeholders to understand the state of the business - in the last resort, to understand whether it is a going concern. One of the key metrics to support this understanding is the profitability of the business.

58 In the long term, profitability is a matter of whether a business makes more money than it spends. There is a formula: profit = revenue - expenditure, which every businessman understands. However, in the short term - over the course of one financial year, or even one quarter - it is more complicated than this. Profit is not simply a matter of cash in versus cash out.

59 Put simply, there are peaks and troughs in cash flow, which need to be ironed out to understand the true

state of the business. One of the simplest examples is the use of a capital asset, such as a computer. The business needs to buy a new computer every five years, and that is expensive. But once done, there is no further purchase needed for another five years. The profitability of the business should not be depressed once every five years, when it needs to buy a computer.

60 Conventions of accounting have been developed, which allow a company's accounts to reflect this reality. The company can 'capitalise' the cost or the computer - so it does not depress profit during the year it is bought - and 'depreciate' that cost uniformly over five years, so that it appears as a uniform cost and depresses profit uniformly over that period. The long-term effect on cumulative profit is neutral. This convention is intended to provide a more realistic picture of the fortunes of the company in each year. But it does introduce an element of management discretion, over which kinds of expenditure are capitalised; and that discretion might be misused for short-term purposes - such as to inflate profits in one year, to attract investors.

61 Other elements of management discretion introduce more subjectivity. For instance, if a company invests some of its resources (money, physical assets, people) in developing a new product, which it hopes to sell profitably over a period of several years, then it may be able to capitalise that investment in the expectation of the profit it will bring later. As before, the long-term effect on cumulative profit is neutral; but year-on-year figures are altered.

62 Other discretionary elements include debtors (where the management needs to assess: how likely they are to pay) and legal disputes (how much money should be set aside in the accounts against an unfavourable outcome?). In all these cases, management discretion affects which accounting period (year or quarter) the profit appears in but does not impact the long-term cumulative profit.

63 All accounting systems, manual or computerised, need to be able to have areas in the accounts where these discretionary management assessments of the business can be recorded, and can be assessed by external stakeholders (such as auditors, acting for shareholders or the taxman) to test whether the assessments can be justified.

64 It is our current opinion that any discretionary elements of the Post Office accounts are not related to the Horizon dispute. The Post Office's business relationships, both with subpostmasters acting as its agents, and with its 'client' organisations such as DVLA, banks or Camelot, all involve flows of cash or assets whose relation to profit is direct and short-term, not involving the management assessments and transient adjustments of the kind we have described above. We have included this discussion in case any discretionary element of PO accounts becomes relevant to the Horizon trial.

3.3 The Users of Accounting Systems

65 From the above it will be evident that information in an accounting system may be of interest to several diverse groups of people, in and around a business. With the onset of Enterprise Resource Planning systems (ERP systems) in the 1990s, which include an accounting system and much other functionality, the scope of accounting systems was widened to include many business functions (such as manufacturing

and human resources, sales and service delivery) not within the scope of a pure accounting system. For organisations, which use ERP systems (such as SAP), the line between users of the accounting functionality and users of the other functionality of the ERP system is blurred. Even if a company uses a pure accounting system (which it typically needs to integrate with other business applications), there are several different classes of user of the accounting system.

66 We first distinguish two main classes of use of an accounting system. An accounting system may be used to provide information to the managers of the business - to help them make decisions about how to manage the business - or it may provide information to external stakeholders such as shareholders or government departments (such as taxation departments) who have a right to information about the business. In the former case, it is known as a management accounting system. In the latter case, it is a financial accounting system. The distinction between the two is by no means clear-cut, in that some accounting systems provide parts of both types of functionality and the information required for the two functions has a high degree of overlap.

67 Users of financial accounting functionality include:

- The corporate finance department
- Senior management, when inputting the discretionary elements of the accounts (often through the finance department)
- External auditors

68 Users of management accounting functionality include:

- Staff in business departments who input the information used by the system (where that information is not automatically generated)
- Managers of 'vertical' slices of the business (such as all PO business with Camelot)
- Managers of 'horizontal' slices of the business (such as a region or branch)
- External auditors

69 Typically, the full scope of management accounts and financial accounts is within scope of an external audit, such as an annual audit of accounts. Thus, auditors need to access both kinds of information.

70 We understand that the financial accounting functionality for the Post Office was provided by SAP over most of the disputed period (from 2004 by POL FS, a SAP application, which in 2010 merged with SAP ADS to form POL SAP), whereas Horizon provided mainly management accounting functionality, as well as Point of Sale functions for the branches. This dual nature required close integration between Horizon and the SAP systems, so that pictures of financial reality given by the two systems were at all times mutually consistent. There is a large overlap between the information stored in the two systems, which will be described below.

3.4 Functionality of Computerised Accounting Systems

71 The functionality of a computerised accounting system includes the following:

* To securely store detailed information about all the assets and liabilities of the company, in a set of accounts, defined by account codes (in the company's chart of accounts)

* To provide facilities for the input and checking of that information, either manually or from other computer systems

* To ensure that the information is always self-consistent, according to the criteria of double entry bookkeeping

* To provide highly flexible output and reporting functionality, which includes:

* Summations of items to verify the self-consistency of the data

* The information required in the annual accounts of the business

* Extracts of the information concerning all transactions made with some external business entity during a period (e.g. the information required to bill that entity, or to pay that entity)

* Extracts of the information relating to all the activities of any internal sub-unit of the business during a period (e.g. a division, or an individual branch), typically used by line managers who have responsibility for some part of the business

* Miscellaneous other slices of information, for diverse purposes (e.g. personnel management), typically used by staff managers who have responsibility for some aspect of the business.

* Many kinds of 'drill-down' and aggregation to inspect individual items in the accounts, or small selected groups of items - for instance, to investigate anomalies

72 Most other functionality consists of extra functions provided by particular accounting systems, rather than by accounting systems in general.

73 From this, it is evident that the functions of an accounting system include the input, secure storage and output of many types of information, in large volumes (many transactions per day), with rather little computation. By far the most important type of computation that occurs in financial accounting is straightforward summation of numerical quantities such as money and stock, subject to the rules of double entry bookkeeping. For management accounting, some other kinds of computation are needed - for instance, for forecasting purposes; but they are usually not computationally demanding or complex.

74 It would be difficult to exaggerate the diversity and flexibility of the reporting and drill-down functionality required of an accounting system. Any person in any management capacity in the business has, through their own particular role, a point of view and a focus of interest about the many transactions done by the business - and that person may require selected 'slices' of information from the accounting system,

tailored and aggregated to the appropriate level for his or her interests. This makes for a substantial number of information slices that may be required from the system. Fortunately, modern computer technology has highly flexible and easily configurable ways of meeting these reporting requirements.

3.5 The Architecture of Accounting Systems - Relational Databases

75 Although computerised accounting systems have been in widespread use since the 1960s, there was a major step change in their architecture in the 1980s, with the advent of relational databases such as Oracle. Hence, it is only necessary to describe the architecture of accounting systems which, like Horizon or SAP or almost every other accounting system now in use, are built on relational databases.

76 The database of an accounting system is managed by a piece of system software called a Relational Database Management System (RDBMS, or DBMS). This is built and supported not by the accounting system developer, but by a supplier of system software, such as Microsoft or Oracle. The accounting software makes calls to the RDBMS to store and retrieve the information.

77 The technology of relational databases matured during the 1980s, and most RDBMS date from that era. They are now very mature, stable and feature-rich products. As they are the essential foundation of accounting systems, it is necessary to say a little here about how they work.

78 The information stored in any computer system can be broadly divided into two types: structured information, and unstructured information.

79 The commonest examples of unstructured information are free text, or pictures, or video clips - of which there are vast amounts in resources such as Facebook or YouTube. There may be some structure which is discernible to a person in a passage of text or an image, but it is not the kind of obvious structure which can be used by a simple computer program. For a computer program to understand and use that kind of structure, it needs to be a very advanced computer program, using techniques of Artificial Intelligence (AI). This is well outside the scope of accounting systems, which use structured information.

80 Two familiar examples of structured information are:

* A spreadsheet

* A bank statement

81 The structure of these is visible in the rows and columns. Each column of a spreadsheet contains only one type of information (such as a number, or a monetary amount, or a date, or a code). Each row of the spreadsheet contains a set of column values (in the 'cells' of the row), which are linked to one another as a single item. The different cells in one row contains different pieces of information about one item.

82 Because of this easily discernible structure, simple computer programs (using no AI) can make use of the information - for instance, summing all the values in one column.

83 While a relational database can sometimes be used to store unstructured information, the main use of a

relational database is to securely store large volumes of structured information. The way it does so can be understood as having large numbers (tens or hundreds) of different spreadsheets (which are called tables) and which are linked to one another. Two different tables in a database are linked to one another (in a 'relation') when they both have one or more columns with the same meaning and share values in those columns.

84 A typical relational database may have tens or hundreds of separate tables; each table may have tens or occasionally hundreds of columns; and a table can have any number of rows, up to millions if necessary.

85 One relational database can act as a 'server' to one or more application programs, which are performing actions directly visible to their users. The application programs make calls (requests) to the relational database management system, which alters or retrieves the data to fulfil the requests. The core service provided by the relational database is to securely store and retrieve this information, for the application programs, or for others who retrieve information from the database more directly. Both the phrases 'securely store' and 'retrieve' have a lot packed into them.

86 The phrase 'securely store' implies several guarantees: that once an item of information has been stored in the database, it will not be lost or unintentionally changed; and that it is a part of a consistent collection of information, whose integrity will not be compromised in any way.

87 The idea of integrity of information is central to relational databases. When the structure (tables and columns) of a relational database is first defined (in a relational schema) that schema defines various types of integrity constraint on the data, for instance that:

- certain columns in tables are mandatory, and will always be given values
- There can never be a record in some table (call it table A) unless there is a corresponding record in some other table B. For instance, there can never be a record of an invoice to a customer, unless a record exists for the same customer. The invoice table and the customer table both have a column 'customer id'; for every invoice record, there must be a customer record with the same customer id.

88 The database management system guarantees that these integrity constraints will be true for all time. If any application tries to make a change to the database which would violate an integrity constraint, that change is rejected by the DBMS, and no change is made at all. One change to the database may involve changes to several tables at once, so that after all the changes are made, the integrity constraints are still true; but part way through the changes, the constraints are temporarily untrue. One such package of changes, involving changes to one or more tables, is called a database 'transaction'. The DBMS guarantees that:

- After any completed transaction, the database will still obey all its integrity constraints.
- After any completed transaction, the changes made to the database can never be lost - even in the event of hardware failures; there are robust ways to recover the information.

- * If a transaction would violate an integrity constraint, it is not allowed by the DBMS; no changes will be made to any table, so the database will still be in a consistent state, as if the change had never been requested.

- * When one application is making multiple changes to the database in a transaction, so that the database is temporarily in an inconsistent state (when some but not all of the changes have been made), those changes are never visible to other applications or users before they have all been completed. Other applications and users can only ever see a consistent state of the database (either before all the changes, or after all changes), in which all the integrity constraints are true.

89 This set of guarantees, given by the DBMS, is called 'transactional integrity'. It has been built into the fabric of all relational databases and has been relied upon by thousands of applications which use relational databases, since the 1980s. Builders of applications can have a very high degree of confidence that their DBMS will maintain transactional integrity. Builders of accounting systems have relied on that guarantee.

90 Selective retrieval and reporting of records is a fundamental capability of all relational databases. All relational databases support the language SQL (Structured Query Language), which can be used from within application programs to retrieve typically small numbers of records from a few linked tables, filtering the records based on the data values in their columns. SQL is a powerful language, enabling an application to pick out from a large database just the few records it is concerned with at any time. The DBMS supports these operations very efficiently, using fast indexed searches rather than brute force inspection of all the records in a table.

91 SQL can be used directly by end users, without the use of any application program, to selectively retrieve and display records. However, it is more common for the users to use a general report writing tool, which not only retrieves the required records, but formats the information in easy-to-read reports - with appropriate column headers, formatting of fields, grouping of records, computations of sums of groups of records, and so on. Nearly all the reports needed from an accounting system are created using these report writing products, which can be rapidly configured to produce any new kind of report, either one-off or regularly as required.

92 All these capabilities of a relational database have been present in relational database products such as Oracle since the 1980s and have been tested by possibly millions of different applications which use those capabilities and rely on them. So, when we are discussing the issue of bugs in an application such as Horizon, the possibility that these bugs arise from bugs in the underlying DBMS - particularly when it is the world market leader Oracle - is extremely remote, and we shall ignore it.

3.6 How Accounting Systems Use Relational Databases

93 Comparing the core functionality of an accounting system - which is to store and retrieve accounting data - with the functionality of relational databases - which is to securely store and retrieve any kind of

structured data - there is a large overlap between them. For many of its required functions, an accounting application program uses the underlying features of the relational database.

94 Therefore, essentially all contemporary accounting systems are built using a relational database, whose database schema has been designed to hold accounting data; and an accounting application program which makes updates to the database, subject to the rules of double entry accounting; and a set of reporting and retrieval functions which are built using the reporting and retrieval functions of the RDBMS.

95 Accounting systems use their underlying database to store information redundantly, so that the effects of each accounting transaction are typically stored in several different forms, in different tables of the database. For instance:

* There is usually some kind of message log, which stores each transaction in its 'incoming' form.

* The accounting information derived from each transaction is stored in both a General Ledger - which holds summary information about all transactions - and several more specific ledgers, which typically hold information at a more detailed level.

* Information is stored in other tables for audit purposes.

96 Given these redundant forms of storing the same information, it is possible to make many checks of the mutual consistency of the different forms. These checks are discussed in the next section.

97 Accounting systems are used by many different types of organisation, which need to track and organise their business in specific ways. To track money and resources in the way best suited to manage its business, each organisation typically has its own chart of accounts, defining a set of account codes, under which all its transactions are accounted for. Each organisation has its own set of rules for allocating income and outgoings to particular account codes.

98 If the accounting application software was written in a way that depended directly on the chart of accounts, then each company would need different accounting software. This would be very uneconomical (in writing and testing all that software), and it is not done that way. The chart of accounts is not 'hard coded' into the application software. The software is written to work with any valid chart of accounts; and the chart of accounts itself is treated as data, stored in the database. In this way, changes in the chart of accounts (which occur from time to time) do not require changes in the accounting software.

99 This is one example of data-driven software - in which any requirement which might change frequently is encoded as data, rather than software code. The code is written and tested to work with all allowed values of the data, to meet a wide range of requirements by changing the data rather than the code. The modern practice of software engineering is to use data-driven software as far as possible. This is not always as far as one might like; some differences between companies and their requirements are best expressed as differences in code, rather than data.

100 Building an accounting application 'on top of' a relational database is one example of a layered software architecture. Most accounting systems, including Horizon, have a layered architecture. At a minimum, three layers are usually found:

- A user interface layer, responsible for presenting information to users, and accepting their inputs
- A business logic layer, responsible for carrying out business processes, and supporting users in doing so
- A data layer, responsible for storing and retrieving data.

101 Most software architectures are more complex than this, with more layers, or with layers within layers. The practice of object-oriented software development, which is now almost universally used for building software applications, encourages the use of many layers.

102 In a pure accounting system, the role of the business logic layer is comparatively simple - in that it does not involve long or complex sequences of business operations. Complex business processes are either manual or are done by other applications. When updating the accounts, the main role of the accounting business logic layer is to accept business transactions, ensuring that they obey the laws of double entry accounting, as reflected in the chart of accounts for the business. Every incoming business transaction is classified as one of several allowed types of transaction. According to the type of transaction, the items within the transaction are mapped onto account codes in the chart of accounts, in a way conformant with the laws of double entry bookkeeping and are passed to the data layer for storage - usually for redundant storage, including for instance audit information.

103 Similarly, for the retrieval and use of the accounting information, the role of the accounting business logic layer is in principle simple, compared to some other applications. Much of the business logic consists of selecting and arranging information that is useful to users such as line managers or the finance department. This is typically done using data-driven report writing software, rather than bespoke application code. The other role of the business logic layer is to carry out many kinds of check, both internal and external, on the accounting data. These checks include the trial balances of double entry bookkeeping.

104 The user interface layer of an accounting system includes the many reports it can produce, and typically includes a modern Graphical User Interface (GUI) used for a wide range of purposes. In the case of Horizon, the user interface also includes a Point of Sale interface, for use in Post Office branches. This interface is best thought of as part of the Horizon Point of Sale application, rather than the Horizon accounting application.

4. CHECKS BUILT INTO ACCOUNTING SYSTEMS

105 Computerised accounting systems are perhaps the most widely used type of computer application in business and are also the most widely relied upon. It is therefore an essential requirement that they should build in sufficient checks on their working to justify this reliance.

4.1 The Double Entry Check, and Other Accounting Equations

106 We first illustrate, in a simplified example, the basic check built into all double entry bookkeeping systems. Suppose the chart of accounts of a company includes, amongst others, two ledgers - a cash ledger, listing various holdings of cash held by the company, and an accounts receivable ledger, listing amounts of money owed to the company by its customers.

107 An initial snapshot of these two ledgers is shown in the table below:

	cash	accounts receivable	sum
	30	100	
	50	20	
	10	15	
		30	
sum	90	165	255

Table 4.1 – Double entry example

108 In practice, each of these ledgers would hold more information. For instance, the accounts receivable ledger would list the identity of the customer who owed each sum of money, the date on which it was due, and so on. They would also typically have many more entries. These details have been left out for simplicity. Each ledger is just a list of items, with no necessary link between the two columns. The fact that some 'cash' item appears in the same row of the table as some 'accounts receivable' item is just an artefact of this simple table and has no significance. The sums at the bottom, as can be verified, are simple arithmetic sums of the columns.

109 Now suppose that the customer who owes £20 pays off £11 of his debt. These £11 are added to the holding of cash which was previously £10, giving £21; and, at the same time, they are subtracted from that customer's outstanding debt of £20, leaving £9. This leaves the ledgers in the state:

	cash	accounts receivable	sum
	30	100	
	50	9	
	21	15	
		30	
sum	101	154	255

Table 4.2 - Double entry example

110 Therefore, the sum of each ledger column is altered - but the sum of the two sums (£255) is not altered. This is because £11 has been added to it, and at the same time £11 has been taken away.

111 The constancy of the sum £255 is the fundamental check of double entry bookkeeping. It can be made at any time, in a trial balance.

112 The double entry check is just one of many accounting equations, in which two separately derived sums must be exactly equal at any time - and therefore the changes to those sums in any time period must also balance. The check is only a check because the data are stored redundantly - with at least two accounting entries for each business transaction. It would be possible not to store the data redundantly, but to calculate each of the matching figures from non-redundant data, in a way that would guarantee their equality. However, that would be single entry accounting, and is no longer done.

113 Similarly, for many other accounting equations, where two figures should always be equal, it would be possible to store the data without redundancy and to compute the figures in a way that was guaranteed to match. That would not be a check on the data. In practice, however, accounting systems store data with a high degree of redundancy so that the checks are meaningful and will detect an error in any one of the redundant data items.

114 This redundant data storage increases the number of independent data items which would need to be altered in a coordinated manner - for instance, by a bug in the software, or by fraudulent activity - to make some change which was not detected by the arithmetic checks.

4.2 Checks on Data Entry, Including Double Entry Checks

115 We note various aspects of how double entry bookkeeping is implemented in a computerised accounting

system, and the checks that are built in:

- In practice, a company's chart of accounts will contain many different account codes (such as for the two ledgers above), and the company will have rules for how each type of business transaction is to be allocated across different account codes. These rules will all respect the rules of double entry accounting, in that each business transaction of any kind must be 'zero sum' in its net effect on all accounts (as seen in the general ledger, when it is balanced).

- In the example above, the item in 'accounts receivable' which changed from £20 to £9 does not have a single date, because it is composed of at least two separate events (the incurring of the debt and paying off a part of it). There is usually some more detailed ledger in which every item is dated, and there is a separate consistency check between the more detailed ledger and the accounts receivable ledger shown here.

- If a discrepancy in the trial balance were to arise, it would be possible to drill down to the most detailed ledgers of time-stamped items to find out exactly when and how it arose. However, other checks in the software make it extremely unlikely to have arisen.

- In a layered software architecture, with a business logic layer and a data layer, there would be double entry bookkeeping checks in both layers. In the business layer, every business transaction could only be packaged up into a set of related postings to accounts (in Horizon's case, a basket) which had zero net effect on all accounts. In the data layer, the software directly above the RDBMS would only accept packages of updates with zero sum and would reject any other package. This practice is known as 'defensive programming' - where the different parts of the software architecture are each built defensively, to protect themselves against possible errors in other parts.

- In this respect, the periodic trial balance of the general ledger is the third line of defence against accounting errors.

- Once a package of updates has been passed to the RDBMS, it may involve several different updates to different ledgers, whose net effect does not alter the balance, but which do alter the balance temporarily when only some of the updates have been done. The transactional integrity of the database guarantees that either all the updates will succeed (so that all of them are securely stored, with no net effect on the balance); or none of the updates will succeed, again with no net effect on the balance. In the latter case, the user will be warned of the failure, and may need to redo some work.

- The database also guarantees that while a sequence of updates is being done for one of its client applications (e.g. for one branch) - so that the database is temporarily an unbalanced state - that unbalanced state is not visible to any other client application. At any time, all client applications can see only a consistent, balanced state of the database.

- * Once accepted, any update to the accounts is guaranteed to be secure and recoverable against nearly all possible hardware errors - certainly it is guaranteed against any single point of failure.

116 The net effect of these measures is a powerful form of error repellency, in the following sense: each business transaction is split into several different postings to the accounts, which obey the double entry zero-sum constraint. This package of updates is created in the business logic layer, is checked to be zero sum in that layer and is delivered to the data layer - where it is checked again. Then the different updates in that business transaction are 'scattered' by the DBMS to many different parts of the accounts - to different tables and rows in the database - and are redundantly copied to other parts of the database. From that point onward, any software error, which could lead to an erroneous computation from any one of those elements, is most unlikely to lead also to a compensating error in the other elements, in other parts of the database. This would be like lightning striking twice, in two precisely coordinated ways - which is vanishingly unlikely. Therefore, any such software error will lead to an error in the trial balance - and it will probably do so frequently. Such an error is very easy to detect.

117 An error in the trial balance is the most serious kind of error in an accounting system. Any such bug should be found and fixed in testing, or at least very early in the service life of the software. So, it could not persist over a long service life.

4.3 Checks in Retrieval and Reporting

118 The trial balance is by no means the only check which applies to an accounting system. The many forms of redundancy built into an accounting database (for instance, between detail accounts and summary accounts, which must tell the same story) all lead to consistency checks - which can be made either automatically in the software, or manually by users, by comparing numbers in different reports from the system. These are all checks in the self-consistency of the data, and many kinds of software bug would lead rapidly to violations of these checks - so the bugs would be quickly detectable and would need to be fixed immediately.

119 The reporting tools used with accounting systems have powerful facilities to 'slice and dice' the data - to produce many different selective subsets of the data, which are vital daily information for many managers. The managers closely inspect these reports, and then 'drill down' and cross-check to find the origin of interesting or suspicious figures. These cross-checks are a part of regular management activity and include both internal and external audit. This constant inspection by many pairs of eyes will soon reveal any software bug which systematically distorts the figures. The accounts matter too much, and matter to too many people, for them to be allowed to be systematically wrong.

120 Some important examples of the cross checks include:

- * **Hierarchical comparisons**: Many large organisations have a hierarchical structure of business units and sub-units (such as divisions, regions, and groups) with managers at each level. Data from the accounting system is a vital tool for the management of this structure, in dialogues between line managers and their supervising managers. This requires hierarchical breakdowns of the figures by business unit, with the figures for each unit being the sum of figures for its constituent units. These figures are the subject of intensive discussions between line managers, and they are often linked to personal remuneration, as incentives. Any inaccuracies in the figures arising from software bugs would be rapidly detected and loudly complained about.

* **Time-slice comparisons**: At any level in the line management hierarchy, managers are expected to understand the time dependency of their financial results - how the full year figures break down into monthly figures, and so on. They rely on various accounting systems to provide all this data and scrutinise their outputs carefully. As a simple example, weekly time-slices of figures may be compared against monthly time slices of the same figures, taken from the same database or from a different database. If the sums do not add up (if a monthly sum does not match the sum of its weekly sums, including part-weeks) questions will be asked. Any underlying software error would be rapidly exposed.

* **Functional slices of the business**: Overlaid on the hierarchy of business units and sub-units may be another functional structure of specialist skills or cost centres such as human resources, marketing, distribution, manufacturing or R&D. This structure has its own management with financial targets and responsibilities. All those 'staff' managers need reports from the accounting system showing the financial performance of their cost centres, over time and in other dimensions. The managers regard controlling these figures as the essence of their jobs, and the figures may be linked to their remuneration, so the figures are all closely watched.

* **Forecast versus actual comparisons**: Any business is required to look ahead rather than backwards, and managers are required to produce plans and forecasts for the business units under their control. All these plans and forecasts are stored in the management accounting system, and their comparison against actual performance is a subject of keen management interest. These comparisons occur through a wide range of reports from the accounting system.

* **Audit checks**: Audits may be carried out for a variety of purposes - external audit for shareholder, regulatory or taxation purposes; internal audits of performance, or audits to detect fraud. For all these purposes, the accounting systems are required to hold extra (redundant) copies of financial information, and comparisons with the extra information are a central part of the audit process. Financial fraud may involve skimming off money in tiny amounts, so these audits must involve very precise comparisons.

121 In summary, diverse types and summations of figures from an accounting system are carefully scrutinised by many people on every working day of the year. If there were systematic errors in the figures from software bugs, these errors would be rapidly noticed, and the bugs would need to be corrected. That is one reason why accounting systems are highly reliable.

4.4 External Checks with Other Organisations

122 So far, we have addressed mainly internal checks, within the organisation, of the consistency and accuracy of its own accounts. There is another important class of checks, which are made between organisations - and are just as unforgiving of any software bugs which would distort the financial picture.

123 Whenever two companies do business together, they need to agree what has been transacted between them - for instance, what goods have been supplied, and what money has been paid. A failure to agree these facts is a serious breakdown of a business relationship, and potentially a breakdown of trust. So, it is important to avoid it whenever possible.

124 In order to know the facts of what has occurred between two businesses, each business relies on its own accounting systems. It does not necessarily trust the other business's accounting systems. For instance, to issue an invoice, company A looks at its own accounting system. To know if an invoice has been paid, company B looks at its own accounting system. A can continue to do business with B only if these two versions of the facts are in agreement. Therefore, the managers involved, in both businesses have a keen interest in knowing that they agree, and in avoiding unnecessary misunderstandings - which would consume management time and reduce trust. The processes for comparing these two versions of the truth, and detecting possible discrepancies, are in all but the simplest cases automated using the accounting systems or data extracted from them.

125 Therefore, for most large businesses, their accounting systems are required to make selective retrievals of the accounts- extracting all the data about their business transactions with some other business - and to compare that data with a corresponding set of data from the other business. Possibly the comparison is made by other applications, using data extracted from the accounting systems. This process of reconciliation, between the versions of truth held by two independent accounting systems, is a powerful check on the accuracy of both versions. Any discrepancy must arise through an error by one or the other party, and it is in both of their interests to find out where the discrepancy arises and correct it as soon as possible.

126 This places an extra premium on accuracy and lack of bugs in both accounting systems. Any bug in either system, which distorted the financial picture of what had occurred between them, would typically lead to repeated discrepancies which, in the absence of any other account for them, would need to be diagnosed and the bug rapidly fixed.

127 This consideration extends a point which we have made previously - that the purpose of an accounting system is to track external reality as accurately as possible, rather than to change it. The true health of the business depends on external reality, outside its accounting system - such as cash, obligations and physical stock - rather than on the contents of its accounting system. Now, however, we must extend the idea of external reality, to include the accounting systems of other businesses that it trades with. In modern financial networks, financial reality is defined by a network of computer applications, and a network of trust which they embody.

4.5 Traceability

128 One of the purposes of an accounting system is to detect financial anomalies; and they have a variety of means, including very flexible reporting and cross-checking. for doing so.

129 As soon as any anomaly is detected - which can be as simple as a figure in some report that a manager does not understand, because it looks too high or too low - it is important to be able to understand the origin of the anomaly. So accounting systems have powerful facilities to drill down, decomposing any figure down to its constituent parts, if necessary down to individual business transactions.

130 Having drilled down to an individual transaction, which may have caused an anomaly in whole or in part, it may be important to answer the question: who or what was responsible for that transaction? This might be a person within the organisation, or an IT system within the organisation, or even some data from a partner organisation. In all these cases, it is essential to be able to trace every business transaction to its originator.

131 Therefore, accounting systems need the means to identify and authenticate every user, and to record the user who initiated each transaction. This usually leads to more redundant copies of data, and more possible cross-checks between them. The need to identify and authenticate users is a necessary protection against fraud.

4.6 Errors Impacting Financial Performance

132 We have so far discussed the general potential for errors in financial data, including errors arising from software bugs. A conclusion of this discussion has been that many classes of software bug would lead to widespread discrepancies in comparisons and in tests with known results, such as the trial balance of double entry bookkeeping. The consequences of these software bugs would be very obvious and intolerable, so they could not last long in the service life of any accounting system.

133 We next discuss errors which are subtler in effect, and do not trigger a rapid alarm such as a failure to balance the accounts. Are there some of these errors which can alter the apparent financial performance of the organisation or parts of it?

134 We say the 'apparent' financial performance, because the actual performance of a business is not defined by numbers in its accounting system - but is defined by external reality such as physical cash and stock, and bank accounts. The purpose of an accounting system is to track that external reality as accurately as possible, using periodic checks (reconciliation) against all types of external reality, to ensure that the tracking remains as close as possible - and does not drift away from reality through a series of errors.

135 If some input error introduces an inaccuracy in the accounting system, that is usually only a temporary inaccuracy. Some later checking process will discover the inaccuracy, and it will need to be corrected.

136 So, for instance, there are many types of erroneous input to an accounting system which can lead to a transient over-statement of profit. When the error is discovered and corrected, there will be a related transient depression of profit, leading to an accurate cumulative profit after the correction.

137 Similar transient effects can apply to any item of the accounts, such as a cash holding. If some collection

of cash is mis-counted in one month; this will lead to an inflated estimate of the cash in hand; but an accurate count in the next month will correct the error, with zero long-term effect.

138 These are errors arising from erroneous input to the system, which naturally obey the principle 'garbage in, garbage out'. It is important to note that although the input is erroneous, it still obeys the principles of double entry accounting, because the accounting software forces it to do so. A balancing double entry is made, but it is an incorrect balancing entry. So, it does not trigger a failure to balance, or any such major alarm.

139 What, then, is the potential for software errors (bugs) which distort the financial performance (even in a transient manner), but do not trigger major alarms such as a failure to balance?

140 In a layered software architecture, there is potential for this kind of bug in the user interface layer. If the user interface displays a cash amount of £3000, while erroneously storing internally a cash amount of £300, and passing £300 to the business logic layer, then:

* The user will think the transaction involves an amount of £3000 (whether that amount is a consequence of his typing it in, or of some automated data capture), and will approve the transaction on that basis.
* The effect on the business logic layer would be exactly as if as if the user had made a mistake, entering £300 rather than £3000.
* This error will get passed on, in double entry, to the data layer and to further retrieval and reporting software.

141 After this error occurs, it could only be corrected by some later checking process - which will inevitably be done. For instance, if the £300 represented a cash amount, later counting of the cash would reveal a discrepancy of £2700. Or if the £300 was a cheque, clearing of the cheque by the bank would reveal the same discrepancy. Whatever the checking process, the net effect would be a temporary upward bump in assets recorded in the accounting system, followed by a later correction to the accurate figure. Because of the checking, the effects of the software error would be transient.

142 Software bugs in the user interface layer have an effect very similar to a user error - causing a transient inaccuracy in the apparent financial performance, which is later corrected.

143 Similarly, there is the potential for software errors in the business logic layer - but mainly only up to the point where the business transaction is split into a set of double entry accounting postings, with zero sum.

144 Up to that point, there is potential for a bug in the business logic layer which converts £300 into £3000. But as soon as the transaction is split into two or more pieces - with £300 going to one account code, and -£300 going to another account code, then it becomes very unlikely for there to be a software bug which converts the £300 to £3000 and makes the same change of -£300 to -£3000[2]. Nearly all software bugs

[2] This kind of bug is very unlikely, but not impossible.

after the splitting into a double entry set of postings would destroy the zero sum, and so the transaction would be immediately rejected - leading to a rapid investigation of the cause of the bug.

145 There is also a further check in the business logic layer. The business logic layer 'dismantles' each business transaction into a set of double entry postings to different accounts. The ways in which it can do so are constrained by the chart of accounts. This chart is defined in data - not in code - so the splitting of the business transaction is almost always defined in data, which drives generic code for all business transactions. The chances of bugs in the generic business logic code are quite remote, since it is exercised and tested by all business transactions. Similarly, the chances of errors in the data defining how each type of transaction is split are remote - because that data is compact and is subject to simple static checking that it obeys the zero-sum accounting constraints; and any other error in the data would lead to robust and reproducible errors for that type of transaction, which would soon be detected in testing.

146 Again, however, even a bug in the business logic layer could only lead to a transient error in the recording of financial performance. Just as for a bug in the user interface layer, some later checking would inevitably take place, leading to a correction.

147 The chances of an error in the data layer which distorted financial performance, and was not rapidly detected, are even more remote.

148 First, the data layer defensively checks any package of postings to accounts, to test if it obeys the zero-sum constraint - and rejects it if it does not.

149 Then, it scatters the different postings to separate parts of the database, with almost nothing to link those postings to one another except their timestamp. The chances of any software error which systematically distorted all these figures, in such a way as not to destroy the trial balance and trigger several other alarms, are very remote. This is not least because the DBMS software has been continually tested by many thousands of applications which rely on it.

150 Next, DBMS software has matured to the point that it offers strong guarantees - for instance, that once a transaction is committed, no data in it will ever be lost.

151 Finally, the data layer makes redundant copies of the data in many different formats - for instance, in a message log of each transaction, and in special forms for audit, and for recovery from hardware failures. There are subsequent tests of the consistency between these forms.

152 Therefore, the chances of bugs in the data layer introducing even transient distortions of financial performance are very small.

153 Finally, we consider the output parts of the accounting system, on the 'other side' of the data layer. Much of this part consists of generic, data-driven reporting tools, which, like the DBMS itself, are relied upon and tested daily by a vast range of organisations, and so are unlikely to have any serious bugs. Configuring these tools is straightforward - much simpler than coding - and any faulty setting up of the

tools will soon be detected by users. The rest of the accounting software on the output side, the part which does need to be coded - is specifically designed for automated checking of consistency. This software could in principle have two kinds of bug:

* bugs which find a discrepancy where there is none (false positives)

* bugs which fail to find an actual discrepancy (false negatives)

154 Bugs of the first kind are very easy to find and correct; they will 'leap out of the page'. Even modest amounts of well-designed testing will find the second kind of bug, by 'planting' discrepancies which the software should find. In conclusion, software for checking is easy to test, and after testing, is unlikely to have serious bugs. If it did, its users would soon detect them.

5. BUSINESS APPLICATIONS IN HORIZON

5.1 Overview of Horizon Requirements

155 The functionality of Horizon is more than that of an accounting system, because Horizon also supports a large, and steadily increasing, number of business applications.

156 For every kind of activity which a customer might enter a post office branch to carry out (such as buying a

book of stamps, or paying a bill, or renewing road fund tax, or withdrawing cash from an account) there needs to be functionality in Horizon, both to support the counter activity of carrying out the transaction, and for the back office activity of settling with the PO's 'client' organisation, who has provided some service to the customer - such as the DVLA, or a bank. Accounting is a thread running through all of these business requirements, but it is only a part of them.

157 The number of services provided by PO branches is large and has increased steadily from 1998 to the present day. The functionality of Horizon has expanded in line with the growth in service, both on the counter and in the back office.

5.2 The Point of Sale Application, and Customer Settlement

158 For part of its activities - such as selling stamps - a Post Office branch acts like a retail outlet, and it needs hardware and software to support this activity. This is the Electronic Point Of Sale Software (EPOSS) component of Horizon. EPOSS must allow the counter staff to record that some goods have been provided to a customer, compute the price of those goods, and allow the customer to pay the money required for all their purchased goods, for instance by cash or a credit card.

159 If a customer wants to carry out two or more different activities in one visit to the counter - for instance, to settle a bill and to buy some stamps - Horizon should not oblige the customer to settle the amount in two separate pieces. So, Horizon has the concept of a customer carrying out a 'basket' of activities and settling the total amount due for the basket in any way they wish - by one credit card transaction, by a cheque, by cash, or by a mixture of these.

160 However, baskets of PO activities and non-PO activities are not supported. If a customer wishes to buy a newspaper and some stamps, the newspaper is not sold by PO - it is sold by a separate retail outlet which uses the same premises. So, the customer has to settle in two parts. In this respect, the National Lottery is an exception and spans the two businesses, as will be described later.

161 So, Horizon needs to support retail-like activities (such as buying stamps) and agency-like activities (such as paying a bill) within a single customer basket, which may be settled by a compound set of payments.

5.3 Agency Activities

162 The Post Office refers to other organisations, for which it provides customer services in its branches, as

its 'clients'. They include for instance high street banks (for offering banking services), gas and electricity companies (for paying bills), DWP (for paying benefits and pensions) and DVLA (for paying road fund tax).

163 The Post Office currently has about 140 client organisations, which shows the diversity of services available in a branch. This also implies that, for most of these clients, the service provided through the Post Office will be different in nature from the service provided for other clients, so some unique software functionality must be provided both in the branch and the back office, to support the activities for that client. This is a part of what makes Horizon such a large and complex system.

164 It is not possible or useful in this report to describe all 140 types of service provided at PO counters, or the software needed to support them. We will only touch on a few services which either illustrate the diversity of services in a representative way or are important in this dispute.

165 A high-level classification of the services now offered in branches on an agency basis includes the following[3]:

* **Paying bills** to BT, utilities, local Government
* **Prepayment services** - DVLA savings stamps, gift vouchers and entertainment tickets
* **Acquiring licences** - local Government permits, television, motor vehicle
* **Money management** - banking deposits and cash withdrawals, savings and investments
* **Insurance services** - general, travel
* **Pensions payments** - e.g. for MoD
* **Lottery** - for Camelot

[3] This list, mainly taken from a 2003 document, will need to be updated.

6. ORIGINAL HORIZON (1998 - 2010)

166 Describing the architecture of Horizon to the court at any point in time, or as it changed over time, presents a practical problem. The problem is: Horizon was so complex, being the result of many thousands of man-years of work, that any description of it, in terms that will help the court determine the Horizon issues, is likely to be long and complex. In our experience and in this case, implemented IT systems are invariably more complex than one would have imagined from the initial requirements. Technology adds complexity. So, there is a long intellectual journey between the evidence disclosed - such as the thousands of Horizon design documents - and what the court needs to determine, which is the expected level of errors in Horizon which might permanently impact branch accounts.

167 The role of the experts is in part to provide a route map and signposts to that journey - so that the court can both follow the journey and validate along the way that the expert signposts are correct and not misleading. Since this expert route involves simplification and abstraction of the actual Horizon, we need to establish that any simplification is appropriate for the court. In places we are forced to say things like: "Horizon is actually complex, but in effect X happens", while also offering evidence that 'in effect X' actually happens. In the expert report, we will need to do this consistent with the court's requirement for the bare minimum of factual evidence.

6.1 Requirements: Branch and Back Office

168 As well as the counter activities described above, Horizon also needs to support the periodic process of balancing and rollover for each branch. Every branch operates in Trading Periods (TP), which are either four or five weeks –according to a timetable published periodically by PO). At the start of each TP, the branch is supposed to be 'in balance'. This means that the physical stock and cash in the branch agrees with the data on stock and cash held in Horizon. Then, during the Trading Period, Horizon records all customer transactions made at the branch, so it records the changes in cash and stock. It also records any replenishments or remittances[4] of cash or stock in the branch. Thus, Horizon records all changes in cash and stock held at the branch during the TP, and can compute, from the starting amounts and the changes, the expected amounts of cash and stock at the end of the period.

169 At the end of each Trading Period, the subpostmaster (SPMR) counts the physical cash and stock in the branch and compares it with Horizon's expectations of the same values. This is called 'balancing'. If the numbers are all equal, the branch is in balance and can 'roll over' to the next period. If the two sets of numbers are not equal, this implies that some of the transactions entered into Horizon during the Trading Period were erroneous or had failed to be entered. For instance, if the counted stock of stamps is less than the expectation from Horizon, this implies that some stamps were given away or lost, without recording a transaction on Horizon. Because it is assumed that this arose through some error by the SPMR or his staff, the SPMR is required to take responsibility for the discrepancy, in some way he or she

[4] At PO, 'remittance' is often abbreviated to 'remming'. This means sending surplus cash from a branch to the centre, or replenishing cash or stock in a branch from the centre.

chooses - for instance, by paying in cash to cover the discrepancy; or by putting the amount in a local suspense account, to be resolved or paid later[5]. Then the branch is again in balance, and can roll over to start the next TP.

170 To support this process, at the end of each TP, Horizon is required to provide the figures of estimated cash and stock; and if the SPMR finds any discrepancy, to enable them to record how the discrepancy will be resolved; and when this has been done, to allow the branch to roll over and start the next TP.

171 Horizon must also support the activities of replenishing stock such as stamps, and of replenishing or remitting cash.

172 It must also support other administrative activities in the branch, such as enrolling new staff and enabling them to use the counter system.

173 The back-office settlement activity of Horizon may be illustrated in the case of a single client organisation, the DVLA. Across the UK in any day, the PO accepts a large amount of money from customers paying their road fund tax. All this money needs to be paid to DVLA. Therefore, PO has a back-office activity - carried out centrally - of summing all these amounts of money and paying DVLA. DVLA knows how much money it expects to receive in this way and checks the amount it expects against the amount calculated by PO. This cross-check is called reconciliation and supporting it and reflecting its outcomes are central to Horizon. Some kinds of reconciliation cannot be done as often as daily, because of variable time lags in the information available to clients.

174 It is important for the court to have an appreciation of the level of complexity of the Horizon requirements, and of the Horizon IT systems built to meet them. In a document ' Fujitsu's Systems and Operational Services to UK Post Office and the Worldwide Trend of Post Offices' Fujitsu have described Horizon as 'Europe's largest non-military IT contract', so Horizon is certainly at the high end of complexity amongst IT systems. It represents many thousands of man-years effort in development and testing, and its documentation alone is of the order of 80,000 documents. Inevitably, the court will only have the time within the course of the trial to understand limited and selected aspects of Horizon. The foundation sections of our report are intended to ensure that the court understands those aspects of Horizon that most need to be understood to address the Horizon issues.

175 On the other hand, compared with the IT estates of various large organisations we have worked for (such as the NHS, Barclays Bank, UBS or RBS), the Horizon system is probably no more complex, and in some ways less complex. The banks' IT estates, like Horizon, have a complex corporate back end and an extensive branch office network. They were developed over a longer time period (30-40 years) using development team sizes similar to or larger than Horizon, often merging together or integrating the IT systems of previously independent organisations. This gave them a degree of legacy complexity, and

[5] The facility for local suspense accounts has not always been available to SPMRs. It was removed for a period of time. We need PO to tell us when.

design compromise, and corporate amnesia, not found in Horizon. There are parts of these IT estates which 'nobody dares touch'. The same is not the case with Horizon.

6.2 The Four-Level Architecture

176 In what follows, the words 'level', 'layer' and 'tier' all have the same meaning.

177 Nearly all complex IT applications are designed in levels or 'layers', to isolate different kinds of complexity in different layers, and to reduce the possibility of unwanted interactions between functions in different layers. A typical 'client server' layering structure includes at least a user interface layer, a business logic layer, and a data layer. The layering in Horizon is more complex than this.

178 The architecture of Horizon up to 2002 is described in the document TD/ARC/001 'Technical Environment Description' which is 476 pages long. This document states: *'The system architecture adopted to meet these requirements is not based on conventional client-server models. Nor does it conform to traditional central-system models. It adopts an entirely original and highly innovative four-tier model that effectively merges the qualities of central systems and client server systems.'*

179 This architecture is explained in a diagram, which appears to have five layers rather than four. If the two boxes of 'Agents' and 'Correspondence' are counted as one 'Agents' layer, we get four layers, of:

* Counter
* Agent
* Host
* External Interface

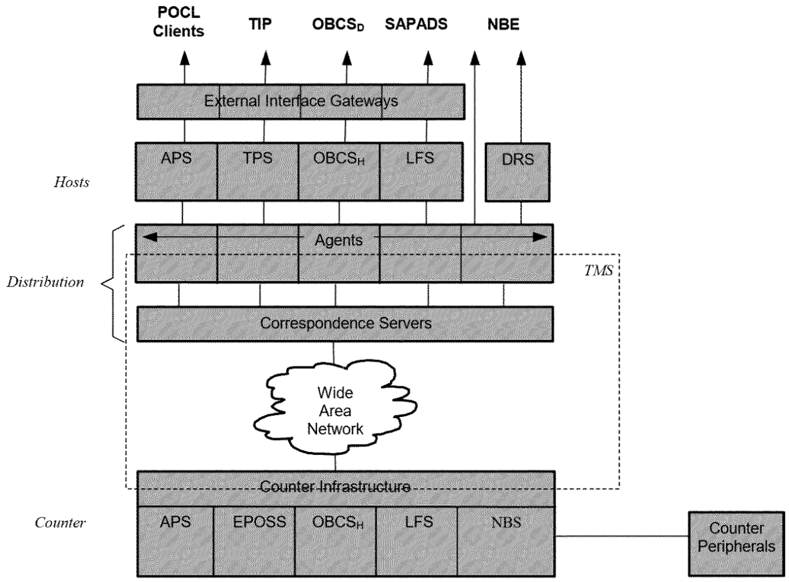


Figure 6.1 - Horizon layered architecture

180 This apparently simple diagram hides a huge amount of complexity in Horizon, and that complexity can only be described in stages.

181 The counter layer consists of all hardware and software in the branch. It includes all hardware and software required to support the counter activities required for all products and customer services offered in the branch. It will be described in the next sub-section 6.3, which will define the acronyms in the Counter layer of the diagram. In this section we will note one aspect of the counter layer: that it was largely built based on a commercial product, Riposte from Escher.

182 Riposte provides the Graphical User Interface (the basis of all user input and output at the counter) and provides a mechanism for secure distribution of messages between the branches and the two back-office campuses, which were located at Bootle and Wigan. This message distribution passed through the Wide Area Network in the diagram.

183 The Correspondence Servers handled communication over the network.

184 The function of the Agent layer was to provide two-way translation of data between the formats used in the counter layer and the network (these formats were described by Attribute Grammars) and the formats used in the Host layer.

185 An Attribute Grammar is a way of describing a tree-like message structure in terms of its parts and their sub-parts. In more modern IT systems, tree-like messages are usually sent in XML (Extensible Message Language), with their structure defined in a notation called XML Schema. This is used in parts of HNG. Because the first Horizon was developed before the use of XML became widespread, Attribute Grammars fulfilled this function in Horizon - we believe because the Escher Riposte product worked in this way at the time.

186 As well as reliable communication, Riposte provided a facility for reliable replication of data between the branches and the back-offices campuses. This means that if certain types of data were created at the branches, it was guaranteed that the same data would be available on the campuses - although if the underlying network was unreliable, it might take some time for Riposte to deliver this guarantee. Replication guaranteed that despite any network failures, no change to data made at a branch would be omitted at the campus or made more than once at the campus.

187 The bulk of the back-office functionality was provided in the Host layer, which will be described in section 6.4. Host applications were and are typically batch systems, processing data in large batches on a daily basis. A complex daily batch schedule was used to control the sequence and timing of these batch processes, using the Maestro scheduling product. The acronyms in the Host layer of the diagram above will be described in that section. It was the Host layer (and for most purposes, only the Host layer) which communicated with the IT systems of PO client organisations, through the External Interface Gateways.

188 There is an important simplification in the four-tier architecture. Each different business application in Horizon (typically tied to a different PO client organisation) can be regarded as a vertical 'slice' though the diagram and is largely independent of the other slices. It is intuitively obvious that different business applications (such as DWP Pensions, and Camelot Lottery) need have very little to do with one another (apart from being able to settle customer payments in the same basket - a facility provided separately from

the applications). Therefore, the apparent complexity of some large Horizon architecture diagrams can be largely ignored when considering a single business application.

6.3 Hardware and Software in the Branches

189 Although the hardware in the branches was not always reliable, and communications in particular were not highly reliable, there were strong measures built into Horizon to ensure that hardware failures and communication failures could not adversely affect branch accounts. These measures are described in section **Error! Reference source not found.**. We shall therefore not spend much time describing the hardware aspects of Horizon, either in the branches or the back-office campuses.

190 In the original Horizon architecture, sufficient data was held persistently in the branches, that a branch could continue to trade, and could support most business applications, even if the wide-area network was unavailable. Whenever the network became available again, Riposte data replication would ensure that the required data became available to the back-office systems. The only applications which could not run in this way were those that required some immediate validation from a client organisation - for instance, withdrawing cash from a bank account. Therefore, a branch was able to hold all the data resulting from a day's trading and more.

191 As will be described in section 7, with HNG this was no longer the case. Persistent data was all stored remotely in the branch database - so that without a working network, a branch could no longer trade. More reliable network infrastructure by 2010 had made this a viable approach.

192 As well as supporting the business applications, the software in the branches needs to support:

- Local user management
- Stock management
- Cash drawer management
- Balancing and reconciliation
- The production of local reports.

193 There had to be sufficient locally-stored data to support all these processes. To keep the counter clerk's view of all these applications consistent and simple, the user interface for all these local applications was provided by the Riposte desktop.

194 To describe how the branch layer of business applications was built on Riposte would involve a lot of technical complexity, most of which would not go to understanding the issues in the trial. We shall instead pick out some aspects which are relevant to the Horizon issues:

- **Zero-sum baskets for customers**: Whatever applications were invoked to serve a customer, the net impact of all the services provided for one customer was a sum of money which the customer was required to settle. It was required that the cash or other money produced by the customer should exactly match the cost of services provided; therefore, the whole basket of services and customer settlement had to be zero-sum, before the basket could be recorded in the branch (and then, through Riposte replication, later recorded in the back-office systems). This was a necessary requirement on all business applications, because the impact of every business application would need at some stage (typically overnight) to be fed into PO's accounting systems, which operated by double entry bookkeeping. The only way to put postings into the accounting system was by double entry, which in turn could only be done for zero-sum baskets.

* **Other branch actions had to be zero-sum**: Any other actions performed in the branches which had an impact on PO accounts (including stock management, cash drawer management, balancing and reconciliation) could only be carried out in the branch in packages of updates which were zerosum, when summed across different PO account codes. This had to be the case, because the only way that the results could be posted to the accounts was to respect double entry bookkeeping - which is zero-sum across the accounts.

* **Transactional integrity**: All branch applications (including all customer business applications, balancing and reconciliation, cash management and stock management) were built so that any zerosum package of updates from those applications would either succeed completely, or would fail completely and have no impact. This transactional integrity was enforced by the Riposte infrastructure. Therefore, it was impossible in any event (such as hardware failure) for a partcompleted set of updates to be recorded in the branch and then replicated to the back-office systems. This was necessary to prevent the accounting system from being subjected to non-zero sum updates, which would violate its double entry basis and cause later failures of its trial balances. The only exception to this principle was the so-called 'recoverable transactions' - where some irreversible interaction with a PO client system took place part way through a transaction - so it could not be undone in the case of a later failure. In these cases, the user on the counter would be guided through a short set of recovery steps, to produce a consistent zero-sum result which reflected what had happened. It was, of course, possible for the user to make some mistake in these steps, which may have been unfamiliar. In these cases, the mistake would be detected later by a reconciliation process, which would typically lead to a transaction correction.

* **Applications driven by reference data**: Many of the business applications were not coded individually but were coded as generic applications which could be configured to run different specific applications by altering reference data. These were referred to in Horizon as 'soft-centred' applications. They had considerable benefits of adaptability and reliability over hard-coded applications (of which there were still a few). New applications can be built and deployed simply by providing reference data, rather than code. Reference data is much more concise and understandable than code, so it is much easier to create it or detect errors in it. Finally, any errors in the underlying generic code would affect a set of specific applications, and so be easy to detect.

195 In the expert report we intend to verify for several business applications and other branch functions that these constraints were actually applied in Horizon.

6.4 Back-End Architecture

196 The three layers of the architecture which resided in the campuses at Wigan and Bootle were the Agent layer (which included the Correspondence layer), the Host layer, and the External Interface layer.

197 As has been described above, the role of the Agent layer was to manage communications and translate data between the representation used in the branches and the network on Riposte, and the representations used in the Host layer.

198 The main design document on the first Horizon[6] says:

'The systems at the Host Layer can provide permanent storage for information if required by the application's business rules. The Host systems can accept data from external Clients, and translate a file-based view of this information into discrete transactions or "messages". These are then passed to the Counters via the Agent and Correspondence Layers. Similarly, messages received from the Counters are translated back into a file-based view for transmission to the external Clients.'

199 Another description in the same document says:

'[Host systems] Servers run mainly large background batch processes and represent the part of the architecture that is responsible for the following functions.

* *Manipulating the information received from the External Client Systems into a form that is appropriate for the presentation mechanism and vice versa*

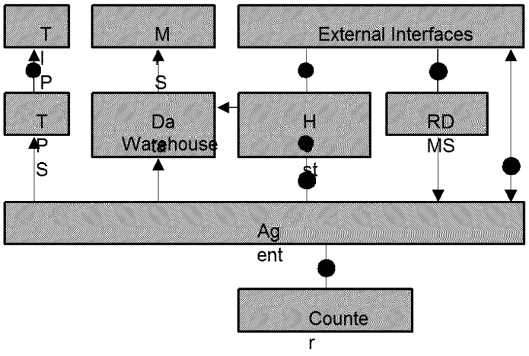
* *Applying business rules that are relevant to that information*

* *Storing non-transient information within the "Data Storage" component. This includes metrics needed for the computation of SLAs that may modify the payments due from PO Ltd for the achievement of key deliverables.*

* *Manipulating any such stored information'*

200 These descriptions only begin to describe the range of functions in the Host layer; to do more, we need to look at specific IT systems in that layer, aligned with different business streams. As will be described in section 7, many of these IT systems did not change with the introduction of HNG in 2010.

201 One fairly simple diagram, which shows a number of important components of the back-office systems, is the following:



[6] TD/ARC/001

Figure 6.2 - Application components

202 It is worth briefly describing those elements of this diagram which have not already been described as part of the four layers introduced in this section:

- * **RDMS** stands for Reference Data Management System. As was described above, many business applications in the branches are driven by reference data, and this approach has many advantages over hard-coding of all the different business applications. It is much more flexible, to manage changes over time and across branches; and it is more reliable. However, this approach implies that the reliability of Horizon depends on the reliability of the reference data (much of which, for instance, is maintained by PO staff rather than by Fujitsu IT staff). Therefore, a dedicated IT application is needed to manage the reference data, and to distribute it appropriately to branches.

- * The **Data Warehouse** consists of one or more databases, whose structure is designed to support flexible and open-ended querying and reporting by PO business staff, to help them understand the whole state of PO business from day to day. Many different kinds of information which pass through the host systems are siphoned off into the data warehouse and stored there in data structures designed for querying and reporting. Functionality which depends on a data warehouse includes MIS (described next) and other applications such as 'data mining' to look for unanticipated trends and correlations in data.

- * **MIS** stands for Management Information System, the component built on the data warehouse to provide PO staff with the flexible access to information about all aspects of PO business. The data warehouse and the MIS are an important part of the checks built into Horizon. In cases of human error in business processes, operational errors in managing PO business on Horizon, or software errors in Horizon, some resulting discrepancy or aberration will be rapidly visible through the MIS. Many pairs of eyes are inspecting the outputs of the MIS, in hundreds of different reports or spreadsheets. One purpose of this is to ensure the rapid detection and correction of many types of errors. These include software errors.

- * **TPS** stands for Transaction Processing System. The purpose of TPS is to 'harvest' all types of transaction taking place in the branches, and to pass them on to other IT systems in the Post Office - initially to TIP, and later to POL FS.

- * **TIP** stands for Transaction Information Processing, which in 2003 (the date of the diagram above) was the gateway to all other PO data processing, including accounting. After 2004, PO accounts were held on a SAP system, POL FS; so TPS passed data to POL FS, rather than to TIP.

- * The black circles denote points *'at which ownership of data conceptually changes and hence at which audit information is generated'*. Audit is addressed in section 6.4 below

203 To quote the 2003 design document: '*One essential task that can only be carried out at the Host layer is*

reconciliation. The Host is the only system component that can detect discrepancies between the transactions carried out at the Counter (and hence reported back to PO Ltd via TPS), and those that were authorised or expected. It should be in a position to send reconciliation reports back to its Client. These enable the discrepancy with the TPS records to be identified and resolved.'

204 This reconciliation, carried out in the Host layer, is an essential element within Horizon for detecting and correcting errors made at the counter. Reconciliation and Transaction Corrections (TCs) are described for both Horizon and HNG in section 8.

205 Reconciliation and TCs, which are primarily intended to correct human errors, also have the effect of detecting and correcting the effects of many possible software errors. If there were any such software error, it would probably occur with such high frequency, and occur uniformly across all branches, giving rise to so many TCs, that the PO would soon suspect a software error (for instance, seeing the effect repeatedly in some MIS report) and require Fujitsu to correct it.

206 The likelihood of any software error in Horizon staying disguised as a human error, and thus of not being detected, is extremely small; and even if it were not detected, by the argument above, it would have no distorting effects on branch accounts.

207 The PO has approximately 140 different client organisations, and so there are up to 140 different types of reconciliation which may be carried out. For the expert report, we intend to examine a sample of the 140 client relationships, which includes some of the largest by volume of money, and many of those involved in the claim. We expect to be able to show that for all these, reconciliation is carried out; so, the error correction method described above does indeed operate. In our opinion, it would be extremely unlikely for any large client organisation to appoint the PO as agents for any other kind of financial transaction such as bill paying, without requiring a reconciliation check that PO was paying them the correct amounts of money. Such a lack of due diligence by the client organisation would not protect the interests of their shareholders or other stakeholders. So, this kind of error detection and correction is used for the vast majority of money that passes through PO branches - for all of its agency business.

208 Host applications fall into one of three classes:

- Complex applications that require a large amount of persistent storage, with high volumes and/or high transaction rates. These generally have their own Oracle database and are located on one of the Host Central Servers.
- Less complex applications, with little persistent storage requirement. These may run on the Host Central Server, or on a Host Ancillary Server, an Intel Platform running under Windows NT Server. Oracle or Microsoft SQL Server can be used to provide the database functionality and storage mechanisms.
- Simple applications that have no requirement for a persistent database may be implemented on a dedicated Intel-based Host Ancillary Server running under Windows NT server. Typically, these generate or process tabular files of text and numbers.

209 We shall mainly consider the first type of host application, which are responsible for the great majority of the money passing through PO branches.

210 Other system diagrams of the Host layer are complex, showing many distinct systems, and there is no 'universal' diagram which is suitable for explaining every issue.

6.5 Audit Information

211 The Horizon system includes an audit database, which is an accurate and immutable record of any activity on any counter which can affect the branch accounts. In the event of any discrepancy arising anywhere in Horizon (for instance, due to a bug in some other Horizon application, or some operational error in running a batch process, or a dispute about what data was entered at the counter) it is possible to compare other records - for instance, records extracted from other applications, or the data warehouse - with the audit records, which are guaranteed to be an accurate record of what was entered into Horizon at the counter. In this way many kinds of error can be traced and corrected. Audit records are normally retained for seven years, as required by the PO contract with SPMs.

212 It is important to understand the many measures used to ensure the integrity of the audit data. A slide set produced by Fujitsu describes this well, and we will summarise the main points here. This can be done through following the sequence of operations by which data travels from the counter to the audit database.

213 In Horizon pre-2010, all data travels from the counter through the software application at the branch, through Riposte data replication to the two campuses, then through the Audit Agent to the Audit Store. This is shown in the diagram:

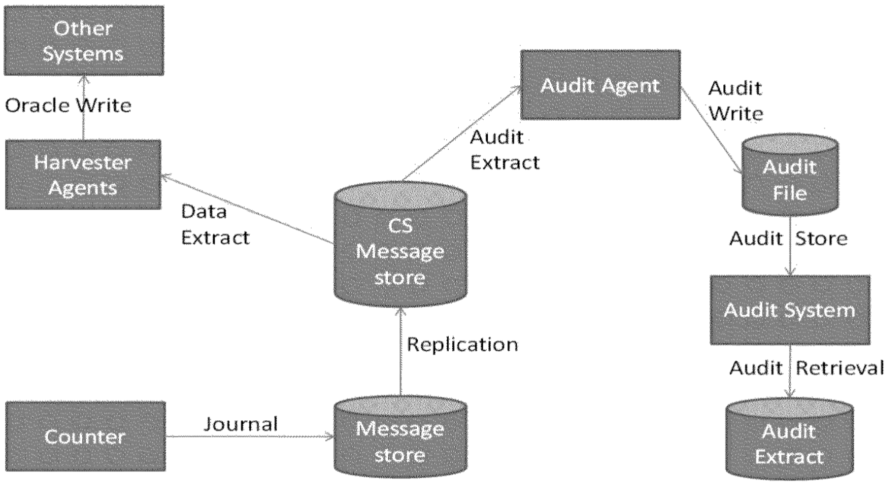


Figure 6.3 - Horizon audit data flow

214 In HNG, all data travels from the counter through the software application at the branch, through communications hardware and software to the Branch Access Layer (BAL), into the Branch Database (BRDB) and then nightly to the audit store.

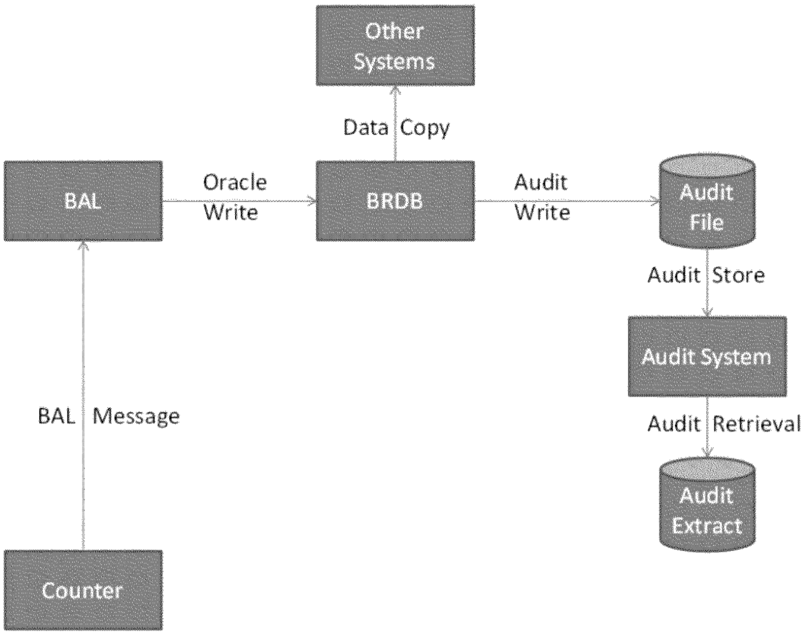


Figure 6.4 - HNG audit data flow

215 The principles ensuring the integrity of the audit data are the same in both cases of Horizon and HNG:

- When a user signs on at a counter, the password he or she provides is used to create cryptographic keys, which are used to encrypt all messages sent over the network to the BRDB, and subsequently used to create digital signatures on the audit records. Thus, any audit record is digitally signed in a way that proves it could only have originated from a certain counter, and that it has not been modified since it left that counter.

- As the counter clerk provides one or more services to a customer, these services and the money paid for them in settlement by the customer are collected in a basket whose monetary sum must be zero. At all stages on the journey of this basket to the audit record, there are checks that it has a zero sum. This is typically not just a check that two numbers are equal and of opposite sign; it is a check that several numbers add up to zero. Thus, any failure in hardware or software, which affects one or more of the numbers, is most likely to destroy the zero sum; if the zero sum survives and the record is stored in the audit database, all the numbers in it are an accurate record of what happened at the counter.

- All baskets are given a journal sequence number (JSN) which must ascend in increments of 1, with no gaps or duplicates. This ensures that no gaps or duplications are introduced in the baskets from any counter, for instance by communications failures or recovery processes. No extra baskets can be introduced without destroying the sequence. All audit entries are time-stamped.

- In communication, data replication, and in storage in any database, principles of transactional integrity are applied. This means that a basket is either stored in its entirety, or no part of it is stored. If it is not stored, appropriate information is sent to the branch, and recovery processes initiated.

- Digital signatures and seals such as MD5 hashes[7] on the audit records ensure they cannot be altered once they have been created.

- Recovery procedures are designed so that should any of these checks fail (e.g. in the event of a hardware failure at the counter), appropriate remedial steps are taken, and the integrity of the audit is preserved.

216 In our opinion, these integrity measures are well designed. For the expert report, we intend to examine system test records to validate that they work as designed and provide evidence that they have worked well over the service life of Horizon. We also intend to establish that the audit records are an accurate record of data entered at the counter in Horizon.

6.6 Changes During the Period 2000 - 2009

217 A series of significant changes were made in Horizon during the period 2000 – 2009. Each new application typically required changes at the branch and at the campuses. Some changes were superseded by later ones. We believe it would be helpful to use a Gantt (or bar) chart to assess the impact of major changes on bugs, which were alleged by the claimants to have impacted branch accounts. Thus far, we have not been able to gather sufficient information from which to construct such a chart. Instead, here is what we have learned to date:

- In 2003 DRS was introduced.

- POL FS came in around 2004.

- In 2005, Pension & Allowance Order Books were replaced by the Post Office Card Account which necessitated building banking services into Horizon. PO had always had business relationships with banks including Girobank.

- AP/ADC was introduced around 2007/2008.

- Around 2010, POL FS and SAP ADS were merged to make POL SAP.

[7] An MD5 hash is a short string, computed from the entire contents of a file, which will change if the contents of the file are changed in any way.

7. HORIZON NEW GENERATION (2010 - PRESENT)

7.1 Motivation for the Move to HNG

218 Horizon moved from the previous Riposte-based architecture to Horizon New Generation (HNG) in 2010. At this time, there was no sudden change in the range of business applications supported by Horizon in the branches. This range of applications has increased continually over the lifetime of Horizon and HNG.

219 There were several motivations for the change from Horizon to HNG. The main driver was to exploit advances in the underlying communication technology, and improvements in its reliability - which meant that it had become possible to store all persistent data at the centre rather than in a branch, with the consequence that a branch could only operate when communications were available - but the risk of failed communications was by then so low as to be acceptable. This change mirrored the wider changes across the IT industry, where increased reliability of communications means that applications can now be 'cloudbased' (entirely dependent on remote data, stored by some cloud provider such as Amazon; and dependent on remote functionality in the cloud) and therefore simpler to deploy and manage.

220 The centralised storage of transaction data allowed several changes and improvements:

- A simplification and rationalisation of the architecture in many respects (just as most cloud-based applications are now simpler than their antecedents);
- Simpler management of the branches in the event of hardware failures or replacements and other events, because in those cases branch data would not be lost and did not need to be recovered;
- No dependence on Riposte data replication, which meant that Riposte could be removed entirely, and all applications could be supported by more modern software technology.

221 These, rather than any change in the business applications to be supported, were the motivations for the move from Horizon to HNG.

222 The document 'Counter Business Architecture' (ARC/APP/ARC/0009) states that:

> *'The objective of the HNG-X programme is to develop a system with structural and operational characteristics that substantially reduce ongoing support and maintenance costs with respect to the current Horizon system.*
>
> *The overall requirement is that the business capabilities offered by the current system (Horizon) are preserved in the new system (HNG-X). However, a limited number of business capabilities will be revised based on a joint optimisation of business requirements and system properties.'*

7.2 The New Division Between Branches and the Back End

223 The fundamental change was that in HNG, no transaction data was held in any persistent form in the branches. The Counter Business Architecture document explains the rationale for this:

> *'The analysis of the serviceability profile for Horizon has highlighted data management as one of the most significant drivers for cost. The storage of transactional data within counters causes the need for security mechanisms that impact*

both the structural complexity and the operational performance of the Counter Business Application. In addition, the presence of sensitive data on the counter increases the time, complexity, and ultimately the cost of maintenance procedures.'

224 In Horizon, on completion of a basket of customer services, that basket was held locally in the branch in the Riposte message store - until Riposte could replicate it to the campuses at Bootle and Wigan, which might have been hours or days later, depending on the state of communications. Meanwhile, the branch could continue to function for many types of transaction. However, the number of applications such as bank withdrawals which required immediate confirmation from a third party, and therefore could not function in the absence of communications, had steadily increased.

225 In HNG, before completion of a basket of customer services, that basket was transmitted and had to be acknowledged by the Branch Database (BRDB). The basket could not complete successfully at the counter until that had happened - so Horizon could not operate in the branches without working communications.

226 Because the branch was no longer responsible for persistent storage or replication of transaction data, the architecture within the branches was simplified.

227 The main difference at the back end was the existence of the Branch Database, which was the main persistent store of all transactions for all branches. Many business applications in the back end were unchanged (and were referred to as 'legacy'), except for the need for them to interface with the BRDB rather than with the previous Agent layer. Other copies of transaction data continued to be stored in those applications.

7.3 New Architecture in the Branches

228 The previous branch architecture had been based on Riposte, which provided functionality on many levels (including for instance user interfaces, some business applications, and message storage and replication). In HNG, Riposte was completely removed; therefore, all elements of the branch software, as well as hardware, were replaced.

229 Whereas much of the Horizon branch code had been written in Visual Basic, for HNG nearly all the branch software was written in Java - a newer language with good support for modern programming paradigms such as object orientation and service-oriented architecture. This allowed a more modern and elegant software architecture in the branch, which did not have to be fitted around the architecture of Riposte.

230 A view of this architecture is shown in the following diagram:

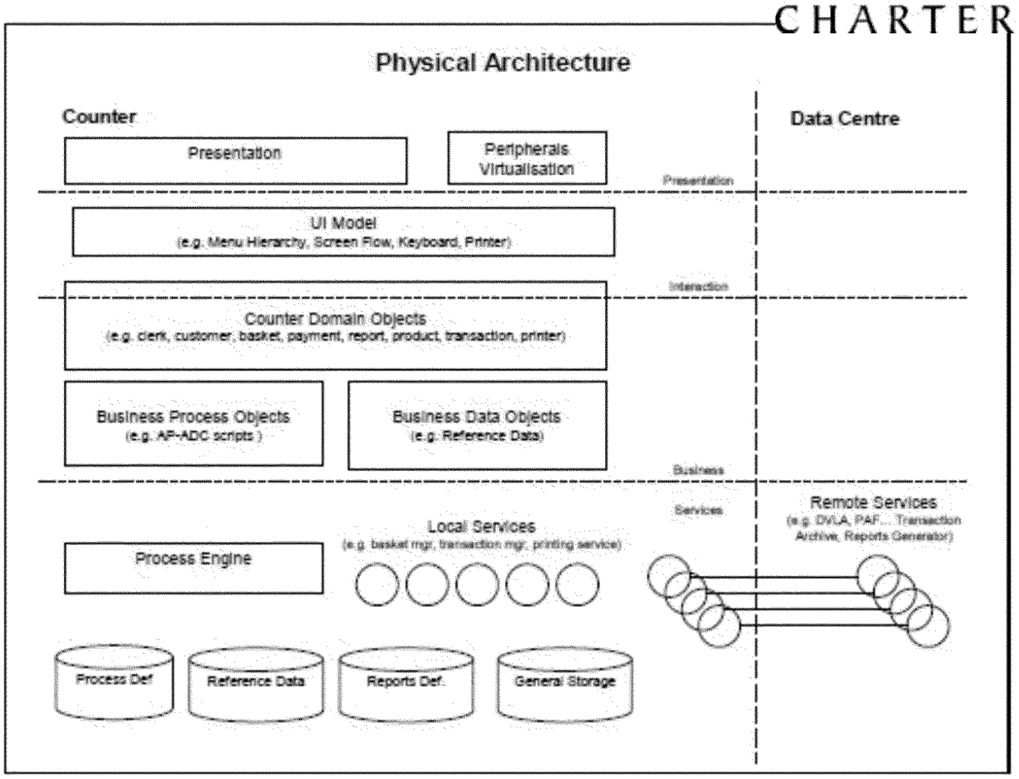


Figure 7.1 - Counter application architecture

231 The top 'Presentation' layer is responsible for displaying information to the users and for accepting their inputs. The next 'Interaction' layer provides the building blocks for this interaction, such as menus. The effect of these two layers is to provide a user interface similar in style to that which had been provided by Riposte in Horizon - to make the user experience similar to what it was in Horizon, but using Java technology, rather than Riposte, to build it.

232 The 'Business' layer provides the functionality of the many business applications, in an object-oriented fashion. This means that there are several general-purpose software objects (i.e. modular blocks of software) with names such as customer, basket and payment, which represent the required behaviour of those entities in the real world, in a way that can be easily reused in many different business applications.

233 The Business Process objects and Business Data objects are more specialised to support the many business applications. As their names imply, the Business Process objects support the sequence of steps which make up a business process, and the Business Data objects hold the necessary data, which is presented at the counter or stored. However, this is generally not done by writing completely different software for each business application (i.e. for each type of service that can be offered to a customer). Many applications are driven by reference data, such as data which defines the sequence of steps in completing each type of service for a customer. This reference data-driven style of software is common modern practice and is effective in making software easier to write and test. New applications can frequently be supported just by adding new reference data, rather than by writing new software.

234 For instance, all automated payment (AP) applications are provided in this reference data-driven manner.

This makes it very easy to build and test a new AP application, for a new client organisation.

235 The Counter Business Architecture summarises the capabilities in the business layer.

> *'In summary the set that are provided by the Counter Business Application are:*

* *Point of Sale Capability;*

* *In / Out Payment Capability;*

* *APOP Facility;*

* *Banking Capability;*

* *DVLA Licensing Capability;*

* *Electronic Top-Up Capability;*

* *Bureau de Change Capability;*

* *Postal Services Capability;*

* *Generic Online Capability;*

* *Payment Management Capability (Cash, Cheque, Vouchers, Debit or Credit Cards);*

* *Cash and Stock Management Capability;*

* *Branch Management Capability (Stock Unit Balancing, Branch accounting, Branch Reports, Reversals and Refunds, Transaction Corrections);*

* *Branch Administration Facility (User Log On / Off, User / Password Management, Stock Unit Creation / Allocation, Provision of Secure Inactivity Time-Out Facilities, Generic User Help System),*

* *Branch Support Facility (Sales Prompts, Bulk Input of transactions, Reference Data, PAF, Message Handling, Audit and Training).'*

236 Just as the Presentation layer does, the Services layer provides a set of software objects which provide services in support of many business applications. Most of these services are not to do with the user interface but help in organising information and sending it for storage in the BRDB.

237 For instance, the facilities for double entry bookkeeping (ensuring that each basket is zero-sum before it is sent) and transactional integrity (ensuring that a basket is either sent and stored in its entirety, or none of it is stored at all) are provided generically in the services layer, and so do not need to be coded individually in the business objects. This design practice helps to ensure that the powerful checks of transactional integrity and double entry bookkeeping are applied universally, and do not have to be built individually into any new business application.

238 One key component of the services layer is the Process Engine. This provides a simplified way for the

counter to provide services which involve a sequence of steps. The sequences of steps need not be defined in Java code but are defined in a specialised Process Definition Language (PDL), which is executed by the Process Engine. PDL was developed for HNG by Fujitsu. The use of PDL means that complex sequences of steps are much simpler to define and test.

239 The disc-shaped boxes in the services layer in the diagram above show that some data are stored persistently on the branch hardware; however, these data do not include customer transaction information. They include business process definitions (definitions of sequences of steps in a process), other reference data, data defining reports that can be output in a branch, and other information required to support operations. The reference data is refreshed daily from the data centre. There are services which provide these data to the other layers in forms that are convenient for them to use.

240 As can be seen from the diagram, the Services layer of the branch architecture is the only layer which communicates with the data centre, through the communications subsystem. Individual services provide reliable and robust communication for various types of information. The purpose, as always, of this layered approach is to provide each kind of functionality (such as reliable and robust communication with the data centre) in one layer only, and not have to reinvent it for many different business applications. In effect, the services layer in HNG now provides many of the services which were formerly provided by Riposte.

241 The services layer also provides interfaces for online services, where to provide some service at the counter, it is necessary to contact some non-PO IT system. These online services include:

- Banking
- Credit / debit cards
- Mobile phone E-Top Ups
- DVLA online
- APOP services, such as postal orders
- PAF lookup
- Generic Online Services
- Some types of PINPad accesses (WSPOS[8]).

7.4 Back-End Architecture: Changed and Unchanged Elements

242 The two completely new elements of the HNG back end are the Branch Access Layer and the Branch Database.

243 The principal function of the Branch Access Layer (BAL) is to exchange messages with the counter

[8] Web Services POS (Point of Sale)

software in the branches. However, the BAL goes well beyond the mere exchange of information, into checking that the information is conformant (for instance, that each basket is zero-sum), logging of all exchanges, and recovery from many kinds of error conditions. Because it has to handle more than 25 million transactions per day, the BAL has many design features to ensure high performance (principally by distributing the load in parallel across many machines), as well as robustness. These design features do not feature prominently in this report but are nevertheless addressed in section 8.

244 The Branch Database (BRDB) is a large, high-performance Oracle database whose main function is to store all customer transactions which originate in any branch. It, too, has many features to ensure high performance and robustness, which will be addressed mainly in section 8.

245 The types of data held in the branch database include:

* Customer transaction data, including both internal counter transactions and external client transactions

* Reference data to be distributed to branches

* Data that applies only to individual branches, such as users, stock units and messages

* Branch report data

* Recovery data

* Journal data

* Postal address data.

246 The logical sub-divisions of the branch database are shown below:

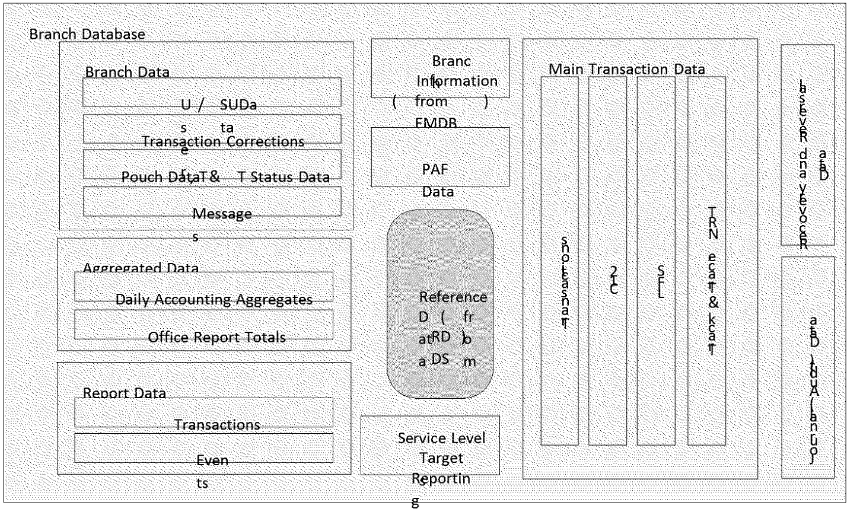


Figure 7.2 - Logical subdivisions of the Branch Database

247 The architecture of the HNG data centre is shown in the next diagram:

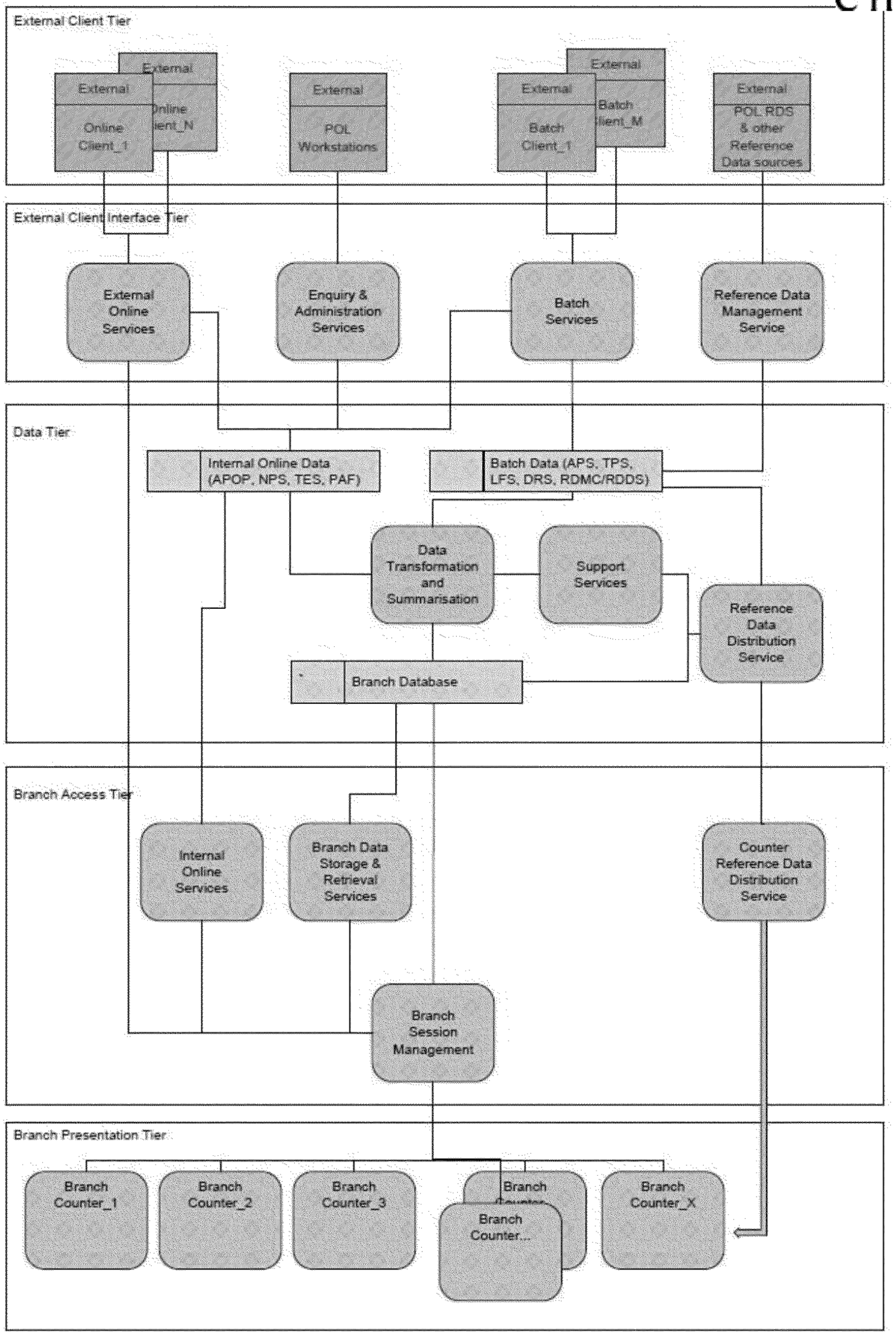

External Client Tier
External
Online Client_N
External
Online Client_1
External
POL Workstations
External
Batch Client_M
External
Batch Client_1
External
POL RDS & other Reference Data sources
External Client Interface Tier
External Online Services
Enquiry & Administration Services
Batch Services
Reference Data Management Service
Data Tier
Internal Online Data (APOP, NPS, TES, PAF)
Batch Data (APS, TPS, LFS, DRS, RDMC/RDDS)
Data Transformation and Summarisation
Support Services
Reference Data Distribution Service
Branch Database
Branch Access Tier
Internal Online Services
Branch Data Storage & Retrieval Services
Counter Reference Data Distribution Service
Branch Session Management
Branch Presentation Tier
Branch Counter_1
Branch Counter_2
Branch Counter_3
Branch Counter...
Branch Counter...
Branch Counter_X

Figure 7.3 - Horizon data centre application architecture

248 From the bottom of this diagram upwards, the Branch Presentation Tier is the branch software, discussed in the previous section. The Branch Access tier (or layer) has been described above, as has the Branch Database.

249 The data tier, which includes the branch database, is a data-oriented view of the business applications, and

other functionality. The External Client Interface Tier provides interfaces in both directions to external client IT systems, pictured in blue in the top layer.

250 Most of the complexity of HNG occurs in the Data Tier and the External Client Interface Tier, which together do all the back-office processing for all the different applications (more than 140 of them) supported in the branches. Because the PO has more than 140 client organisations, and each one of them may have differing requirements for back-office processing such as settlement and reconciliation (depending on their own differing IT systems), there are at least 140 kinds of back office processing to be supported. While many of these have strong similarities between them, the differences between client organisations cannot be entirely removed by the External Client Interface Tier; so, much of the complexity and diversity of these applications resides in the Data Tier. Many of these applications are batch applications, harvesting transaction data from the BRDB and running once per day in a complex batch schedule.

251 The documentation provided by Fujitsu includes many different 'wiring diagrams' of these back-office applications - each from a slightly different viewpoint, emphasising some aspects and abstracting out , or omitting, others. Because the full picture is so complex (with probably more than 140 boxes, and many more lines between them), it is very hard to provide the court with simplified and useful views which may not need to be revised and amplified later for specific purposes. However, the following data-oriented diagram gives one useful view:

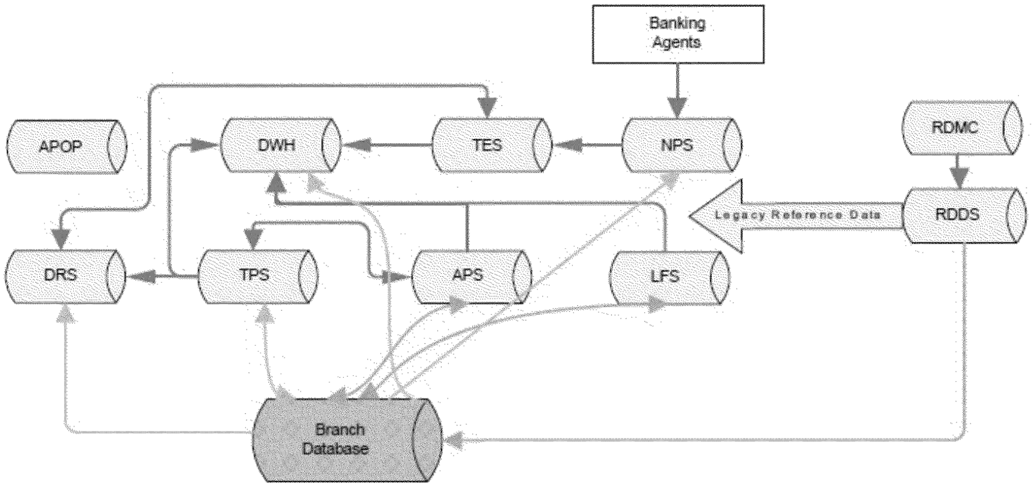


Figure 7.4 - Application Database Architecture

252 This diagram shows the central role of the Branch Database (shown in pink) and a number of so-called 'legacy databases' shown in pale pink which survived unchanged from the previous Horizon. The acronyms for the legacy databases are as follows:

- * APOP: Automated Payment Out-pay Database
- * APS: Automated Payment Service
- * DRS: Data Reconciliation Service

* DWH: Data Warehouse

* LFS: Logistics Feeder Service

* TES: Transaction Enquiry Service

* TPS: Transaction Processing Service

* RDMC: Reference Data Management

* RDDS: Reference Data Delivery Service

253 There are of course omissions from this diagram - including, for instance, the Audit database, which continued in its previous role as described in the section 6.5 above, storing the same data in the same form as before, but now taking its information from the branch database.

8. ARCHITECTURAL TOPICS ACROSS HORIZON AND HNG

8.1 User Error Detection and Prevention

254 Horizon processes something of the order of 6 million counter transactions per day. Therefore, even with highly trained and careful users, it is to be expected that there will be several thousand user errors per day, spread across all the branches. It would be surprising if the rate of user errors in customer transactions was less than 1 in 10,000 - which would lead to 2,500 user errors per day.

255 It is therefore essential that Horizon should be designed to minimise the level of user errors which occur, and that it should correct them whenever possible. In addition, Horizon needs to be proof against any user error which might benefit a subpostmaster (fraud), either alone or in collusion with a customer. If a user error can be detected immediately, that implies that it can be prevented; and it generally will be. Therefore, 'error correction' inevitably means 'error correction sometime after the event'.

256 In the design of the Horizon counter user interface, there are large numbers of measures to prevent user errors. Many of these measures have by now simply become common practice in the design of user interfaces - such as the use of menus and buttons, rather than free text input, to allow the user at any time only to choose one of the actions or inputs which are allowed at that time, or the use of facilities for inputting numerical values, which only accept numbers (not characters) in an allowed range, and confirmation buttons to ensure that the user really intended to take the action he chose.

257 Among these measures is the check that when a customer makes one or more payments for a basket of items or services, the sum of the payments entered must be the same as the summed cost of the items purchased; the basket cannot be concluded unless it is zero-sum. Also, in all possible cases (such as credit card payments) capture of the amount paid is automatic rather than manual, preventing any user error.

258 Despite these measures, there remain cases where the amount of cash entered into Horizon (which gives a balancing basket) is not the amount actually put in the till; or where the amount of stock given out, such as stamps, is not the same as that entered in Horizon. There is in principle no way in which Horizon could detect or prevent these kinds of user errors. They are errors made outside Horizon, in the handling of cash or stock. So, they can only be caught by later error correction measures. These measures are powerful and, in the absence of later errors made in the correction process itself, will always eventually correct these sorts of user error. The delay involved in 'eventually' will be discussed below.

259 The first form of error correction is the regular checking of cash and stock made by the subpostmaster. If the wrong amount of cash is put in the till, a later check of the cash will reveal the discrepancy - which must then be corrected by putting cash in or taking cash out of the till - similarly for stock such as stamps. This means that, if there is no error in the later checking process, any errors in handling cash or stock during the day will be exactly corrected, leaving Horizon with an accurate picture of cash and stock. Multiple errors can be corrected in a single accurate check.

260 There are cases involving external clients which cannot be corrected by this check, but which can be

corrected later by a reconciliation process. Suppose a customer comes to pay a £50 gas bill, but only offers £30 cash; and the counter clerk does not spot this. There are three possible cases:

a) The counter clerk enters into Horizon that a £50 gas bill has been paid, but only enters £30 as tendered by the customer. This error is prevented immediately, because the basket will not balance to zero.

b) The counter clerk enters that a £50 bill has been paid, but only puts £30 in the till. This error will be found later by the daily check of cash - and the subpostmaster will lose the £20 he has to put in to correct it. The customer saves £20.

c) The counter clerk enters that a £30 bill has been paid and enters £30 as tendered by the customer. This error is not prevented by the zero-sum check or found by the daily cash check; but it will be corrected later by reconciliation.

261 In case (c), the Post office will report to the gas company that a £30 bill has been paid and will pay the gas

company a sum for that bill (and others). The gas company will know that the bill they issued to this customer was £50, not £30 - so will realise that they are £20 short. They will require an extra £20 from the Post Office, who in turn will issue a transaction correction, which in effect requires the subpostmaster to cover the loss. He will be able to see which day the bill was paid, and to see from Horizon that his desk clerk recorded an amount of £30 - while the gas company can provide evidence that the amount of the bill was £50. Again, the subpostmaster has to find £20 to correct the error, just as in case (b).

262 The extent of the delay in case (c) depends on the agreement between the Post Office and the gas company - in particular how often they carry out reconciliation - and on any further internal delays in the Transaction Correction process. However, there are cases similar to (c) where the delay is inevitably longer.

263 Suppose, as case (d), that the customer pays a £50 bill with £50 cash. In the rather unlikely event that the desk clerk puts the £50 in the till, but neglects to enter the bill payment at all into that customer's basket (so that any other items in the basket still balance), then, in his next check of cash, the subpostmaster will find that he is £50 'ahead' and may take out the cash. Nothing has been entered in Horizon and Post Office makes no payment to the gas company. The gas company will simply think that the customer has not yet paid his bill.

264 It is only much later, when the gas company starts chasing this customer, and he says: "But I paid that bill at the Post Office; and (possibly) I have evidence to prove it" that the gas company realises that they are owed £50 not by their customer, but by the Post Office. The gas company demands £50 from the Post Office; and, when the customer has identified which branch he paid the bill at, the Post Office issues a transaction correction against that branch. The net result is that the subpostmaster, who at one time thought he was £50 ahead, ends up exactly in balance, and his accounts on Horizon are correct again.

However, this process may take many months, depending on how long the gas company gives its customers to pay their bills.

265 Through these measures, essentially all errors in entering amounts of cash or stock in daily trading are prevented or are corrected after some delay. It is hard to think of any such error in counter transactions which is not caught in one of these ways. This is necessary, because, as we have seen, there are probably several thousand such errors made at the counter every day.

266 User errors made in stock taking or monthly balancing have a similar effect; they get corrected eventually. Suppose, during monthly balancing, the subpostmaster mis-counts some item of stock; so, for instance, he thinks the stock is in balance with Horizon, when it is not. Horizon thinks that the stock is in balance, with a physical stock of X units. But in fact, the physical stock is Y units. The subpostmaster should have counted it as Y units, and then made up the discrepancy (X-Y) in cash; but he did not. Then, over the next month, the changes in stock (as recorded in Horizon, and in fact) are Z units. At the end of the month, Horizon thinks that the stock should be (X+Z) units. But the physical stock is actually (Y + Z) units. So, the discrepancy, which is (X-Y) units, still has to be made up at the end of the next month. The effect of a user error in balancing in one month is just to postpone the need to balance, for another month. After balancing correctly in that month, the accounts will be accurate again.

267 In effect, users know that making an error in balancing in their own favour in one month will not get them 'off the hook' for making it good in the next month. They can postpone a problem, but they cannot make it go away. This is essential to avoid a class of deliberate user errors.

268 There are possible user errors in recovery situations. In practice, because these situations occur more rarely than typical counter transactions, and are often more complex and unfamiliar to them, they are more prone to user errors.

269 For instance, recovery is necessary when there is a hardware failure or communication failure in the middle of a basket of customer transactions. In HNG, because customer baskets are sent to the BRDB in one 'all or nothing' success unit, the usual effect of a hardware failure in the middle of a basket is no effect on the BRDB at all, so there is nothing to recover from; when the hardware is working again, if the customer is still there, the desk clerk merely has to start the whole basket again. The effect of hardware failure on Horizon was similar. Of course, when the failure occurs, the desk clerk must know whether any cash from the customer has been put in the till; and if the customer does not want to wait until the hardware is recovered, must give it back to him.

270 This raises two possibilities which must be taken care of:

* A hardware failure occurs at such a time that the desk clerk does not know whether a customer basket has been completed.

* A hardware failure occurs before a basket has been completed, but when some irreversible interaction with an external party (such as a credit card payment) has been made.

271 The first case is addressed by Horizon, after hardware recovery, making it clear to the desk clerk which

was the last basket to complete, so he can carry on appropriately. If he has completed a basket, but Horizon has not registered it, he needs to re-enter it. Consider what happens if the clerk misunderstands this information, and thinks a transaction was completed and sent to the BRDB, when it was not; or if there is a long delay before recovery and the transaction is not re-entered. For simplicity, consider a sale of stamps. This means a physical transaction has taken place (cash in, stamps out) whereas it has not been recorded in Horizon. Compared with physical reality, Horizon's stock of cash will be low, and its stock of stamps will be high. This means that at the monthly balancing and rollover, the subpostmaster will find these two discrepancies and will have to make them good, at no net cost to himself.

272 The second case is known as a 'recoverable transaction', because some action is needed to bring the transaction to a consistent state. A recoverable transaction occurs when some irreversible interaction with an external agency, such as an authorisation of a payment by a bank, occurs at some stage during a customer basket, and the basket later fails for some reason (such as a hardware or communication failure). Then typically some action is later required from the counter staff to 'recover' the transaction to a consistent state. User errors may occur during this recovery process - which is less familiar than the normal operation of Horizon. In these cases, typically the error is caught later in a reconciliation with the external party and is corrected by a TC.

273 User errors may also occur when making changes to stock in the presence of no customer, such as 'remming in' or 'remming out' (short for 'remitting in' or 'remitting out') cash or stock. For example, consider remming in cash that has been sent to the branch. The Post Office has a separate record of how much cash was sent. Consider when the clerk enters the wrong amount; £2000 was sent by PO, but the clerk entered £1000 (while putting £2000 in the till). This means that Horizon's record of cash in the branch is £1000 less than physical reality, which would allow the subpostmaster to take out £1000 at the next balancing and rollover. However, probably before that (because it depends only on PO's internal processes), there will be a reconciliation which detects that while some other part of PO sent £2000, the branch only remmed in £1000, This leads to a transaction correction which both alerts the subpostmaster to the mistake and puts Horizon's record of cash back in line with reality. If this happens before the next balancing and rollover, it saves any further cash movement at rollover; if it occurs after, the subpostmaster can take out £1000 which he will later have to repay.

274 So, because of reconciliation, the long-term effect on branch accounts of any user errors in stock operations are zero. This is of course necessary, to prevent any deliberate errors which would fraudulently benefit the user. However, the effect of some errors can be to require the subpostmaster, in order to roll over to the next trading period, to effectively lend money to the Post Office. This can happen if, near the end of a trading period, the user remits in £3000 but mistakenly records it twice. Then, Horizon records £6000 remitted in, whereas physically only £3000 have been added to branch cash.

8.2 Intrinsic Error Prevention

275 'Intrinsic error prevention' is a broad term, because it includes any design feature of the Horizon code

whose effect is to make the occurrence and persistence of serious software errors less likely. In this definition, we need to consider the levels of severity of software error, considering factors such as:

* whether the error affects only the immediate user experience, or whether it has any effects on stored data;

* whether it can affect the accuracy of branch accounts, and if so whether temporarily or permanently;

* whether its effects are so severe and obvious that it will inevitably be rapidly spotted and corrected;

* how frequently it is likely to occur.

276 Generally, we are considering only the most serious errors - those which can have a long-term effect on branch accounts, which can occur with significant frequency, and which can remain undetected for long periods of time.

277 Intrinsic error prevention includes the following techniques:

a) Double entry bookkeeping, which ensures that any numerical error affecting only one part of an accounting transaction will destroy a trial balance and be rapidly detected.

b) Transactional integrity, which ensures that in many cases, partial updates to databases, which would destroy their integrity and consistency, cannot happen.

c) Measures designed to detect or correct user errors, which, as described in the previous section, in many cases also have the effect of detecting or correcting software errors. These are so important that they have been described separately in section 8.2.3.

d) Defensive programming, where small parts of a program are written to assume that other parts of the program may be in error and are written to always check their inputs for the presence of errors.

e) Redundant storage of data, where the same information is stored repeatedly and in different forms in distinct parts of the IT estate, with consistency checks on versions of the same data. These checks include arithmetic checks of monetary sums.

f) The audit system provides a highly secure and tamper-proof record of what is entered into Horizon at the counter, which can be used, in cases of any anomaly, to provide a 'gold standard' for comparison with data held in other parts of the Horizon estate, supporting the diagnosis of software errors.

g) Data-driven programming, where specific functionality is achieved by generic software modules, driven by reference data: such generic modules are simpler to code and easier to test, and the reference data is easier to manage and is less error-prone, than software code. Errors in the generic code would have such widespread effects as to be rapidly detected and corrected.

h) Software coding standards, to ensure consistency of work by different developers and to discourage coding techniques which are more error-prone.

278 It is our understanding, from reading the extensive documentation of Horizon and HNG, that all these techniques have been widely and consistently applied across the whole Horizon IT estate. To catalogue the many ways in which all the techniques have been used across Horizon would be a very lengthy exercise, and in our view would not assist the court.

279 We shall discuss each of the topics (a) - (h) in the following sub-sections, illustrating where it is applied in Horizon rather than listing all its applications; then we shall later apply the topics to the analysis of specific bugs.

8.2.1 Double Entry Bookkeeping

280 The double entry bookkeeping check has been described with examples in section 4.1. Wherever double entry bookkeeping is used to store any accounting information, it implies that every financial transaction is split, at an early stage of its journey through the IT systems, into separate monetary amounts (accounting postings) whose sum should be zero. In the simplest case, there are two postings of amounts +X and -X; but there may equally be three postings X, Y and Z which sum to zero. The different postings then take different routes through the system - whether they are similar parallel routes into the same tables of a database, or widely divergent routes to different databases.

281 This means that any software error affecting the finances (i.e. an error which would have an effect on branch accounts) is likely to have different effects on the different postings. Typically, a software error will affect one type of posting, but not another. This means the sum of the postings in a basket will no longer be zero, and the sum of all postings will be changed. Errors which do not affect the zero sum (e.g. doubling all postings in a zero-sum set) may occur but are in practice rare. Some errors can cause the same zero-sum set of postings to be made twice or not at all - which will not destroy the trial balance - but this type of error is usually detected by the measures for detection of user errors. We intend to address this more fully in our expert report.

282 Accounting systems such as POL FS sometimes require that account postings are put to them in zero-sum sets (this is an example of defensive programming, described below); but more important, in all cases they from time to time perform 'trial balances' to ensure that all postings that have been put to them, in whatever time order, have had a zero sum since the last trial balance. A wide class of software errors, which might affect branch accounts, would destroy the trial balance. A failure to balance the accounts is always treated as a serious condition, so any software bug which led to it would have to be diagnosed and corrected very rapidly. The vast majority of such bugs are found in testing and never make their way into live use.

283 The double entry or zero-sum constraint was applied widely in Horizon and HNG. In the HNG counter software and the Horizon counter software, any customer basket, made up of any mix of products, was required to be zero sum, and this check was made at several places in the counter software. It was also made in the HNG Branch Access Layer before entry into the BRDB; and finally, postings from the

BRDB into POL FS had to be zero sum (were not allowed to destroy the trial balance) We intend to verify this for our expert report. Similar constraints applied to many types of non-customer operation, such as replenishment of stock or monthly balancing; although, as we shall see in section 9, they did not apply to all such operations.

8.2.2 Transactional Integrity

284 The transactional integrity constraint has been described, in the context of accounting systems, in section 3.5. Because transactional integrity is a fundamental facility built into all database management software, and it is necessary, for any relational database, to describe in its schema the integrity constraints which it must obey at all times, we know that transactional integrity was applied to all of the many databases of financial information in the Horizon system - including the BRDB, the POL FS database, and many others.

285 This means that any compound package of updates, applied to any of these databases, would have been applied as a single transaction or 'success unit' which would either completely succeed, or completely fail leaving no trace. It would be impossible to leave any of these databases in an inconsistent state, not satisfying its integrity constraints.

286 Transactional integrity gave protection against a wide variety of conditions:

* Hardware errors or communication failures

* Software errors which led to failures and cancellations

* Users deciding to cancel some operation when it is half-way through.

287 The use of transactional integrity at the database level makes it much easier to write software which will recover correctly from a wide range of conditions such as these.

8.2.3 Measures to Correct User Errors, which also Cancel the Effects of Software Errors

288 In section 8.1 we have described how many classes of user error are detected and then corrected by stock counting or reconciliation processes, so that there is no permanent error introduced in branch accounts by those user errors.

289 The same design features - introduced in Horizon for correcting user errors - also cancel the effects of a large class of possible software errors.

290 Consider for instance a software error whose effect was to lose a whole basket of customer transactions, while making it appear to the counter user that the basket had been fully processed. The effect of this software error would be identical to the user error of carrying out a physical transaction with a customer - such as selling stamps - and neglecting to enter it into Horizon at all. The previous two checks - double entry accounting and transactional integrity - would not catch this software error. However, since its effects are identical with those of a user error, the measures which ultimately cancel out the effects of the user error, would also cancel out the effects of the software error.

291 In this case, the regular check of stock against physical stock would reveal two discrepancies - one of

stamps, and one of cash. In order to achieve balance and roll over, the subpostmaster would have to make good both discrepancies, which he can do at no cost to himself. The final effect is that the accounts on Horizon are accurate - and the software error has not adversely affected the subpostmaster.

292 Similarly, a software error which resulted in a basket of postings being stored twice would resemble the user error of entering the same basket twice - and its effects would be later cancelled by the same mechanism.

8.2.4 Defensive Programming

293 It is a universal modern software engineering practice to write programs defensively. This means to design a program as a set of small software modules, making the interfaces between the modules as simple as possible, with each module expecting the inputs it receives from other 'sending' modules to obey certain constraints - such as the double entry bookkeeping constraint of zero-sum baskets. The sending module is built so that its outputs should always obey those constraints, and it is tested to ensure that its outputs obey those constraints.

294 However, in a belt-and-braces approach, the receiving module should not trust the sending module - but should where possible check that its inputs obey the constraints it is expecting to be obeyed and should automatically raise an alarm if they do not. The receiving module is tested with sets of invalid inputs, to ensure that it really does raise the alarm. Then, raising the alarm is treated as a serious condition, which requires immediate diagnosis of the error and re-testing of the source module - which is usually done in testing, before any live use. If this is done, then any failures in the design, coding or testing of the sending module are detected by the receiving module.

295 This technique has become especially powerful with the use of object-oriented programming, which encourages and supports design in terms of small software modules (objects) with well-defined interfaces between them.

296 The result is that as the scale and complexity of IT applications has grown, the number of serious bugs which survive testing or persist in live use does not grow linearly with the number of lines of code - because although the number of bugs initially written in to the code may grow with the number of lines of code, the number of checks which detect and expose those bugs also grows, and may even grow faster; so the number of serious bugs which survive beyond integration testing does not grow. This good practice has been essential as the complexity of IT systems has grown in recent years.

8.2.5 Redundant Storage of Data

297 In a complex organisation such as the Post Office, there are large numbers of staff and managers with different, but overlapping, responsibilities for various parts of the business. For instance, for each of the Post Office's many client organisations, there may be a manager within the Post Office with responsibility for all the services offered in branches on behalf of that client; and there may be staff reporting to that manager. There may be supervisory managers with responsibility for groups of client organisations, and

there may be staff managers with responsibility for aspects of the business across many client organisations.

298 To carry out their responsibilities, all these managers and staff require different and overlapping views of PO's business information, including that originating from Horizon. They require regular financial reports generated at different intervals, concerning different and overlapping slices of PO's business operations.

299 This means that the data generated at the Horizon counter must flow not only to the BRDB and to the central accounting system, POL FS. It must also flow to other databases and data warehouses used to generate other reports. Furthermore, as IT systems are usually, for technical reasons, more complex than a layman would expect from understanding their requirements, there is a larger number of different stores for the same data - in many different slices and representations - than one would at first expect from PO's business needs.

300 Because there are many redundant copies of the same data, it becomes possible to carry out automatic checks that these redundant copies of data are consistent with one another. In a simple example, one database may hold daily summaries of some financial or stock information, while another database holds weekly summaries. These two summaries may have reached those databases by different routes through the organisation and its IT estate. However, there can be a simple and powerful arithmetic check that the weekly summaries are consistent with the daily summaries. It is common good practice to build the IT systems to make these checks wherever possible, and to raise an alarm if any check fails. Failure of a check may arise from some software error, or from a user error, or from an IT operational error such as failing to run some daily batch process. Whatever the cause, it needs to be diagnosed quickly - because until it is fixed, some of the reports from those data will not be useful to managers - being known to be inaccurate.

301 There is a further check on the correctness of the data, in that different managers all look at the reports they receive and have many discussions about the content of those reports. If any error in the data leads to regular inconsistencies between the different managers' views of the business, those inconsistencies are usually soon revealed and must be corrected.

8.2.6 The Audit System

302 We have described a situation in which data about some customer transaction in a branch, having reached the BRDB, then fans out (typically through several 'harvester' programs) to many different IT applications and databases in the back office.

303 As has been described in section 6.5, the audit sub-system of Horizon holds a reliable and tamper-proof record of all accounting transactions initiated at the counter. So, in the case of any discrepancy between two or more of the many other databases and systems which comprise Horizon, the conflicting versions can each be compared with the audit system record, which can serve as a reliable record of what was entered at the counter.

304 This means that any software error occurring in any of the systems in Horizon, which has an effect on

branch accounts, will lead to a discrepancy between that system and the audit system. For a software error, the same kind of discrepancy will probably occur on many occasions, and on each occasion can be investigated by a comparison with audit data. The audit data makes it easier to isolate and correct software errors in any other Horizon system - particularly those errors which might affect branch accounts.

305 Comparison with the audit system may also help in detecting data errors which may have arisen in other ways, such as:

* an error in running one of the many daily batch processes;

* a user error by some member of PO's back office staff; or

* any tampering with branch accounting data.

306 The audit system gives protection against tampering with branch account data, because audit records are signed with digital signatures dependent on a password known only to branch counter staff.

8.2.7 Data-Driven Programming

307 Another common modern software engineering practice which has been applied in Horizon is data-driven programming. The data in question is often referred to in Horizon as reference data (although there are also other kinds of reference data).

308 For instance, as described in section 6.3, data-driven programming has been referred to in the old Horizon
desktop software as 'soft-centred' applications. As a second example, section 7.3 describes the Services layer of HNG which contains a general 'process engine', driven by reference data in the form of PDL (Process Definition Language), which provides a straightforward way of supporting different business processes at the counter.

309 The effect of this data-driven approach that, instead of having to write specific software to handle different uses cases, the developer writes generic software which is driven by different reference data for different use cases. This improves the reliability of software in three ways:

* The generic software is often simpler that the specific software which would have to be written to support different uses cases; therefore, it is less prone to errors.

* The generic software is tested by applying it to all the different use cases; therefore, the generic software is more thoroughly tested, and less likely to contain undetected errors.

* The reference data, which must be supplied for each use case, is much simpler and easier to read and understand than the code which would otherwise be written, and is often the subject of static validation checks, which are easily made. Therefore, any errors in the reference data are easily detected and corrected.

8.2.8 Software Coding Standards

310 The use of software coding standards is mentioned in section 8.7 on development and testing of Horizon. It is also covered here because of its impact on the likely level of serious errors in Horizon.

311 If programmers are allowed to develop software, each in their own preferred style, then each one may use different programming techniques, with the result that they may have difficulty understanding the code that other people have written. It is a practical necessity for programmers to understand, extend and modify each other's work - so lack of mutual understanding of code would be harmful. Therefore, most organisations apply software coding standards to ensure consistency and mutual understanding of code.

312 These coding standards develop over time in such a way as to discourage programming styles which are more error-prone. They may include recommendations such as defensive programming, and other forms of checking - which will reduce the level of undetected errors.

8.2.9 Summary of Intrinsic Error Prevention Methods

313 In our expert report we intend to show from our analysis of the Horizon design, coding and of its development standards, and of Fujitsu's test records, that as Horizon has grown, the expected level of serious software errors, as defined above, has not increased but rather has decreased. We expect to find this because the number of checks implied by (a) - (h) has grown faster than the number of lines of code. If the number of checks grows faster than the number of possible software errors (which is roughly proportional to the number of lines of code), then the number of errors which escape all checks does not increase, but rather decreases to a very low level

314 The test of this conclusion is the in-service record of Horizon. The claimants' expert has, by examining the Known Error Log and other sources, found evidence for a number of specific bugs which have occurred during the in-service life of Horizon and HNG, and we have conducted similar searches. In our expert report we shall analyse those bugs according to criteria described under heading (a) - (h), to assess how many of them could have had permanent effects on branch accounts. The result of this analysis will be the main conclusion of our expert report.

8.3 Financial Audit

315 A core requirement of any accounting system is to support the activity of annual financial auditing, which is required in order that external stakeholders in the organisation, such as shareholders, can have confidence in the published annual accounts. Confidence in the annual accounts has a number of dimensions - including checks that any discretionary element to the accounts, which might move profit from one year to another, have been fairly applied by the officers of the company. Audit also includes checks that the financial accounts are an accurate reflection of business activities, and that the accounts could not have been altered by any fraudulent activity.

316 So, the annual financial audit may be regarded as just another management cross-check on the figures in

the accounting system - but for a large organisation like the PO, it needs to be a particularly careful and thorough one. Auditors need to be aware of how inaccuracies, poor controls, adverse financial events or fraud might be covered up or hidden in the accounts, and to examine those areas with particular care.

317 Constraints of time and resources may mean they can only do this on a selective basis - but even when auditors work selectively, they must be empowered to drill down to any level of detail, and the accounting system and the auditing process must give them an ability to do so. They need to be able to retrieve any figures they choose, analyse them as necessary, and interview managers about them. They need to be able to do this outside the control of the organisation itself. This places large demands on the ability of an accounting system to produce many different views and subsets of the accounting data - and any one of those views may then be subject to the consistency check of discussions between the auditors and the management.

318 During much of the life of Horizon, the PO accounting system has been POL FS (later known as POL SAP), both built on the SAP ERP system. SAP is a very mature and capable ERP system, with highly mature and capable accounting facilities and reporting facilities built into it. It is entirely capable of supporting a probing and independent financial audit process. At all times, the financial data within POL FS about branch operations has come from Horizon or HNG. Had those systems been providing erroneous or unreliable data to POL FS, it would have been the business of the financial auditors to draw attention to this fact. Our expert report may include limited factual evidence about financial audit.

8.4 Reconciliation, Transaction Corrections and Transaction Acknowledgements

319 The claim and the defence have drawn attention to transaction corrections. This section briefly describes how they operate and how they usually arise as from a reconciliation operation.

320 Whenever the PO acts as an agent for some external client organisation, financial transactions take place for which both the PO and the client organisation have a record. The PO's record of the transaction starts at some branch counter on Horizon. The client's record of the transaction may be available to the client immediately - as in the case of a cash withdrawal from a bank - or it may only be possible to link the client's record with the PO record after some delay. This happens, for instance, where customers pay bills at a PO branch. Initially, the client organisation issues a bill to a customer - so knows the amount of the bill. The customer then pays the bill at a PO branch - after which the client organisation may check that the amount paid was as billed.

321 Whatever the delay involved, there is eventually a process of comparing a client organisation's record of events taking place at branches, with the PO's own records. This process is called reconciliation, and it is done on a transaction by transaction basis. There are many differences of detail in how reconciliation is carried out for different client organisations, or where it is carried out; sometimes the client organisation does it from a file sent it by the PO, and sometimes the PO does it.

322 However it is done, and wherever it is done, the result of reconciliation is always in principle the same. For the vast majority of transactions carried out (several million per day), the client's record of a transaction and the PO's record match exactly, and there is nothing more to be done. However, for a small minority of transactions (which is typically a few thousand per day) there is some mismatch, which needs to be investigated and corrected. How this is done may depend on the terms of the contract between the PO and the client organisation.

323 When reconciliation finds a transaction for which the PO record and the client record do not match, it is passed to a department in PO accounts which handles reconciliation discrepancies. Each such discrepancy is, until it has been dealt with, an error in the accounts - and so it must be dealt with. The task of this department is to determine how each discrepancy arose, which therefore how it needs to be dealt with.

There is only a small number of ways in which a discrepancy may have arisen:

* It may have arisen though a mistake in the client organisation - in which case, the client organisation will need to correct its accounts, taking a loss if necessary. When the client organisation's systems are entirely automated (as, for instance, with a bank) this is extremely unlikely to occur - and the PO would need to have good grounds for believing it arose from a client mistake.

* It may have arisen from some mistake made in the back office of the PO, which would include a bug in Horizon or a failure in running some batch process. Since these processes are highly automated, it is likely that any such failure, if it occurred, would produce a series of reconciliation failures with some very distinctive pattern, which would be easily recognised as such and soon corrected. While this was being done, it would be necessary to correct the PO accounts by adjusting the figures of some central department, which would not affect the accounts of any branch.

* The most likely cause is that the discrepancy arose through some manual error in the branch. In this case, the correction of the discrepancy must be made in the branch accounts, unless the subpostmaster can show in some way that it did not arise in the branch or was not his or her responsibility. The process for doing this is a transaction correction.

324 When the appropriate department in PO decides that responsibility for a discrepancy lies with the branch, a request for a transaction correction is issued and is passed from POL FS to the BRDB. At this point, there is no impact on branch accounts. The request is passed on from the BRDB to the branch Horizon system, so that it will show on the subpostmaster's screen when he starts the Horizon system the next morning. At this point, the subpostmaster may either accept the correction or may dispute it and ask for further investigation.

325 It is only when the subpostmaster has accepted a transaction correction that it enters his branch accounts, and therefore enters the audit sub-system in a record sealed with his own password.

326 Transaction corrections may arise for reasons other than reconciliation discrepancies. For instance, when a

subpostmaster recognises that his accounts need correction (for instance, after mistakenly remming in the same amount of cash twice), he may request a transaction correction.

327 Transaction Acknowledgements (TAs) occur more frequently than Transaction Corrections, because they occur through normal branch business, in the absence of any errors.

328 For instance, when a customer pays a bill at a Paystation, the Paystation terminal transmits amounts to Ingenico, who inform the PO each day what the transaction totals were for each branch. This results in a TA being sent to the branch. When the TA is accepted by the subpostmaster, it enters the branch accounts to balance the cash from the Paystation.

329 Similar processes apply to Camelot, because lottery terminals are not directly connected to Horizon, and operate outside Post Office business hours.

330 Thus, Transaction Acknowledgements are used to balance branch accounts for cash received at a branch which is not automatically entered into Horizon at the time it is received - because the cash amount is recorded on a separate device not connected to Horizon.

331 There are cases where the TA circuit through Ingenico does not work, because of connectivity issues. In those cases, the subpostmaster can find out the cash amount from a printed summary from the Paystation and request a manual TC to balance the cash with the branch accounts. Without either a TA or a TC, the branch would have a cash surplus.

8.5 Hardware and Software Resilience

332 The Post Office's branch business depends heavily on Horizon. One of its design principles is that the system should support non-stop trading during core business hours. Therefore, it is important that the system operate successfully even when its computer hardware, software or networks fail. The ability of an IT system to protect users from any type of disruption and to maintain acceptable service levels is known as 'resilience'.

333 Resilience is required in all the major components of Horizon in branches, data centres and networks:

- Hardware
- System software, such as the DBMS and communications products
- Horizon infrastructure and applications software
- Networks

334 '*A single point of failure (SPOF) is a risk posed by a flaw in the design, implementation or configuration of a system in which one fault or malfunction causes an entire system to stop operating.*' [9] One strategy for minimising the impact of a failure is to replicate major components of the system. Error processing and recovery procedures also improve resilience.

335 This section of the report will introduce the range of measures adopted by Post Office to make Horizon

[9] https://searchdatacenter.techtarget.com/definition/Single-point-of-failure-SPOF

resilient. These measures reduce the risk that any system failure impacts on branch accounts.

336 Several specific qualities of the system, which can be viewed as aspects of resilience, are addressed in other
sections of this report:

- Recovery from failure is discussed under Intrinsic Error Prevention - section 8.2

- Transactional Integrity – section 8.2.2

- Security – section 8.6.

8.5.1 Branch Hardware

337 Each branch uses a standard PC at each counter. These PCs run the software that provides Horizon functionality to the SPMR and their staff.

338 Many branches have more than one counter and the failure of one (for example through equipment malfunction or loss of power) does not prevent the use of other counter systems.

339 In the original Horizon system, PCs installed in branches with just one counter were fitted with exchangeable hard disk units, as well as fixed disks. If the PC failed, the information held on the external storage could be moved to a replacement PC.

340 Each PC is equipped with peripherals such as a touch screen, a customised keyboard, bar code reader and printer. Peripherals can be replaced more easily than the PC itself. Workarounds are also available for many of the devices:

- Keyboard - many transactions are driven entirely from the touch screen or other peripherals and require no use of the keyboard. The software can display a simplified 'soft' keyboard. The clerk can touch the relevant area on the screen to simulate pressing the associated key.

- Bar code reader - all transactions that use data from the reader also allow it to be manually entered by the clerk.

- Printer – the PC software includes a 'print preview' facility for all reports including receipts. The clerk can use this and then copy the screen contents manually onto paper. This is a slow process, but it does provide fallback if necessary.

341 The hardware failure most likely to have an impact on branch accounts is the loss of a PC. This loss may be temporary, in which case the PC is simply restarted. All the data that had been secured prior to the failure remains available for use.

342 In the original Horizon system, important data was stored locally on each counter's PC. Using the Riposte software (described above), this data was replicated across counters - to the exchangeable disk in a single counter's PC - and to the Correspondence Servers in the Horizon data centre.

343 If a PC could not be restarted and was replaced, the data needed to continue trading was retrieved from

one of the replicated copies. In this way, the new PC could continue where the old one left off.

344 On HNG, all the important data for each counter is secured centrally. Therefore, a replacement PC simply connects to the data centre and continues as if the previous PC had been restarted (e.g. establishing the working data storage needed to service customer transactions and support the rest of the branch's business operations).

345 The procedures for recovery from PC failures depend on the user. If they are in any doubt as to whether a transaction has been completed to the failure, they can use the transaction logs to confirm the position before taking any further actions. If any transactions are lost, the user should re-enter them.

346 In exceptional circumstances, there is the possibility of a transient impact on branch accounts. For instance, in the original Horizon, if there was a network failure followed by a PC failure, there was a slight risk that transactions in the intervening period could have been lost. Similarly, there were issues in single counter branches if the PC failed and was replaced before it had replicated to the Correspondence Server.

347 The impact of such cases should only have been temporary because the principles of double entry accounting, transactional integrity and accounting checks built into the system (and described above) would have detected and thereby prevented any long-term discrepancies.

8.5.2 Data Centre Hardware

348 The counter systems used in branches are supported by data centres known as campuses. Each campus contains many substantial hardware platforms. In the interests of simplicity, these use the minimum number of different equipment and operating system types.

349 In Horizon, the platforms supported servers and gateways, PO corporate systems, and management and help desk systems. HNG upgraded the servers and storage devices, introducing newer technologies.

350 For resilience, both generations of Horizon have relied upon two physically secure campuses. Originally these were located at Wigan and Bootle, with each providing fallback for the other. The campuses had a similar network configuration and servers. Each, and all the equipment in it, was sized to be capable of running the entire workload. The simplified diagram below illustrates the replication and redundancy of major components that help to avoid SPOFs:

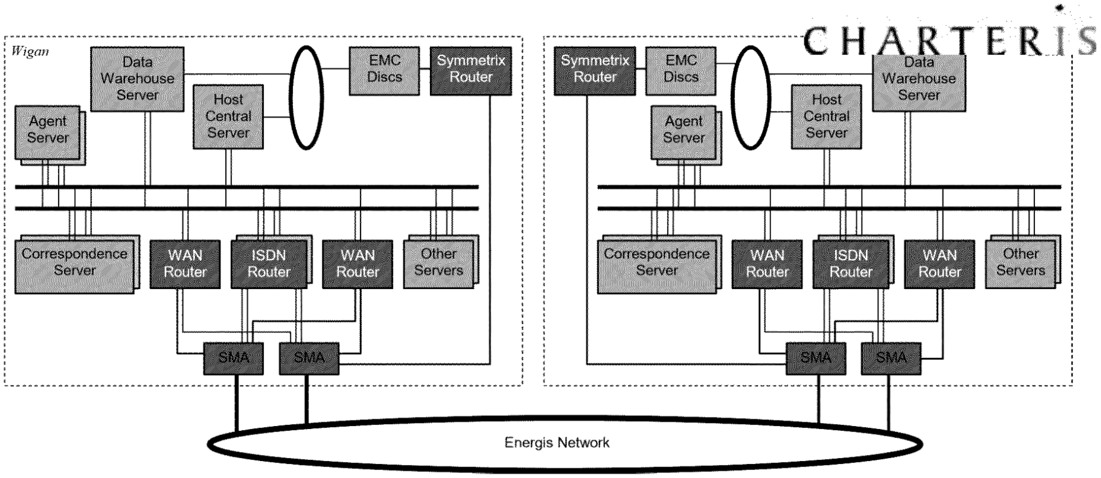


Figure 8.1 – Redundancy of major Horizon system components across campuses

351 Disk mirroring is a technique used to protect a computer system from loss of data due to disk failures. It is a form of backup in which anything that is written to a disk array (on a single site) is simultaneously written to a second disk. If one hard drive fails, the data can be retrieved from the other mirrored hard drives.[10]

352 The resilience strategy for the Host Central Servers (shown in Figure 8.) involves replicating data between the two sites. Thus, following a site failure, the servers at the other site can quickly take over the entire workload. A single disk array is used in each campus. Each array supports internal disk mirroring, with automated recovery from the mirror in the event of a single disk failure.

353 In HNG, both data centres have been relocated to Belfast. One campus supports the live operation while the other provides disaster recovery (DR). Under usual operation, the DR Data Centre is used for testing. Some 'Live' elements of the solution are operational at the DR Data Centre where this is required to support DR.

354 Each data centre can support the entire branch business and is configured so that no single failure leads to loss of service. Data is replicated from the Live Data Centre to the DR Data Centre to ensure that, in the event of disaster, there is:

* No loss of transactions received from the Branch estate where those transactions have been committed to the Branch database.

* No loss of the audit trail

355 Switchover from the Live Data Centre to the DR Data Centre is manually initiated.

8.5.3 Branch Software

356 Fujitsu has selected mainstream infrastructure products such as Windows, Unix and Oracle[11]. These

[10] Description based on https://www.techopedia.com/definition/25959/disk-mirroring

[11] Unix and Oracle are used on servers at the data centres, rather than in branches.

products lead the market, because they have proven to be reliable. As far as practicable they tolerate faults, thereby providing a degree of resilience to the solution.

357 The software running within each branch is now quite different to what was included in the first Horizon system prior to the year 2010.

358 In original Horizon, the key software was Riposte. This provided the user interface and messaging infrastructure.

359 Counter clerks rely mainly on EPOSS, which allows them to record that some goods have been provided to a customer, calculate prices and accept payments.

360 As explained more fully in section 8.5.1, resilience of the branch PC is supported via replication (and automatic recovery) of transaction data across nodes in the network.

361 One of the Horizon architecture documents includes the following principle: '*Applications should be designed and built defensively, so that they can handle any type of unexpected conditions in a controlled manner*' [12]. In other words, Horizon software has been built to detect errors and respond appropriately – rather than fail. Any exceptions in lower level components are trapped and handled within the calling software. Errors are logged, with event notifications directed to support staff so that they can analyse the problems encountered. In our experience, there is a limit to the application of this worthy principle. As error handling is an overhead that can adversely affect performance as well as development costs, it should be included in software judiciously.

8.5.4 **Data Centre Software**

362 As we have explained earlier in this report, Oracle databases and the associated tools have been widely used across Horizon. They provide a resilient platform for the most important applications. Riposte components running on servers also formed part of the picture for tolerating and recovering from failures.

363 In 2001, Oracle introduced Real Application Cluster (RAC). A cluster is a set of connected nodes (computers used as servers) that work together so closely together that they can be viewed as a single system. RAC allows cluster of computers to run the DBMS software simultaneously while accessing a single database. By 2006, in the original Horizon system, RAC had been used to provide resilience in the Network Banking Service but the software is more central to HNG.

364 The most crucial component of HNG is the Branch Database (BRDB). Without it, branches cannot trade. Therefore, its design must support non-stop trading during core hours.

365 BRDB is built on RAC with a four-node cluster, which provides high availability. Each node has four processors, which also provides high performance. The database has no single point of failure. Multiple nodes also reduce the probability that a set of simultaneous failures can cause a complete loss of service. If one node fails, the remaining ones carry on running and the database remains available for use. A

[12] Technical Environment Description for the original Horizon system (TD/ARC/001), section 11.6.3.3

standby database is maintained automatically; this allows very fast recovery if a fault takes the live database offline. A disaster recovery site remotely mirrors the data. The mirroring of data is synchronous. This guarantees that no data is lost if there is a catastrophic site failure.

366 The connection from the counters to the Branch Database is through the Branch Access Layer (BAL) as shown in the diagram below:

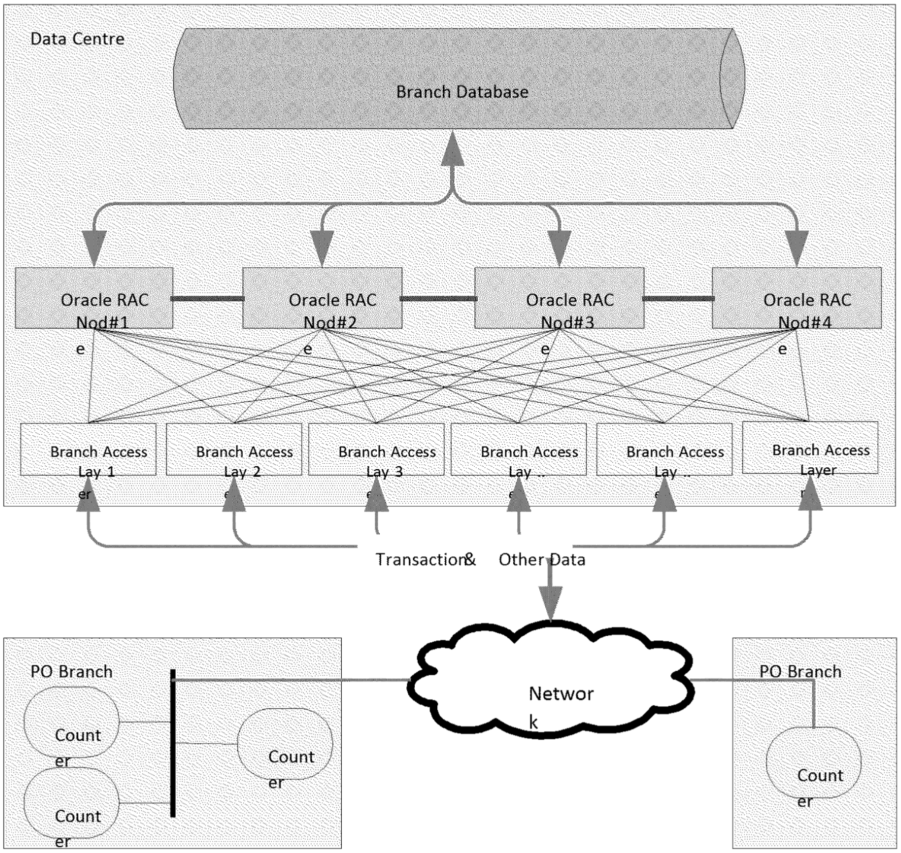


Figure 8.2 - Oracle RAC in the tiered architecture

367 The resilience of application software is underpinned by isolating components across the tiered architecture (as described in section 6.2 above), by the resilience of the platforms supporting them and by the discipline of defensive programming.

8.5.5 **Networks**

368 Computer networks are built using a combination of hardware and software elements. Examples include gateways, routers, firewalls and virtual private networks (VPNs).

369 The following diagram provides an overall view of the Horizon networks:

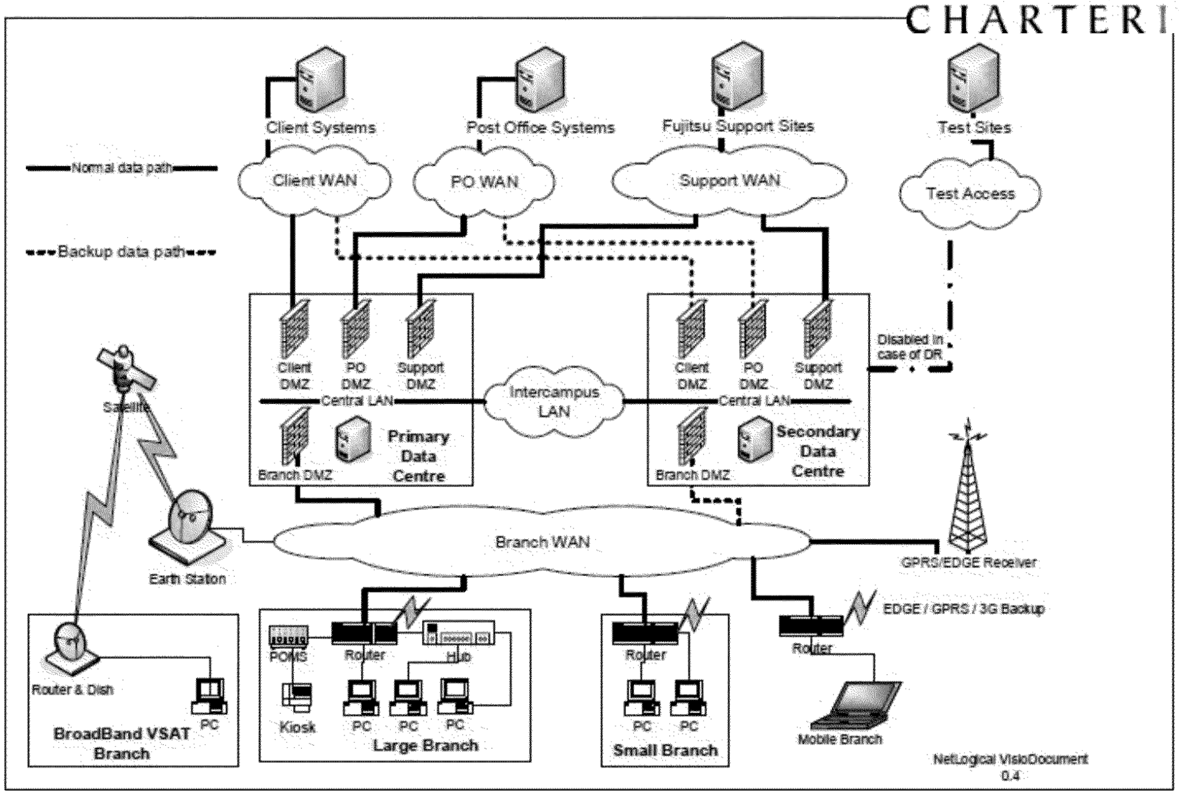


Figure 8.3 - Horizon network overview

370 In the original Horizon, the WAN was built on ISDN[13] connections, although a VPN service was also used between the branches and campuses. Branches could continue to trade with no WAN partly because the connections at that time were less reliable. This feature of the architecture reduced the need for resilience across the Horizon system.

371 By 2006 many of the ISDN connections had been upgraded to ADSL[14]. Branches located too far from an exchange use VSAT[15] satellite technology and mobile branches used EDGE[16], GPRS[17] or 3G[17]. EDGE/GPRS/3G is also used as a backup for ADSL.

372 The communications infrastructure carries networking services over several types of connection with different bandwidth, depending on the needs of the services using that link. Resilience is improved by replication of most networking components with alternative routes being provided, where appropriate, in case a primary route fails. The branches have network connections that can use either data centre. Either both data centres are connected (using more sophisticated connections) or one data centre is connected

[13] The standards for Integrated Services Digital Networks were first defined in 1988.

[14] Asymmetric Digital Subscriber Line

[15] Very Small Aperture Terminal

[16] Enhanced Data rates for GSM Evolution: GSM stands for Global System for Mobile telecommunications [17]

General Packet Radio Service

[17] Third generation of wireless mobile telecommunications technology

normally with the branch able to use the other data centre if there is a problem. The VPN software in the counter connects to multiple VPN servers at both sites to provide resilient encrypted tunnels.

373 The PCs in each branch are connected with each other and to peripherals using a Local Area Network (LAN). One of the local PCs is designated as the gateway through which the branch systems communicate with the rest of Horizon.

374 A Wide Area Network (WAN) connects branches with the two campuses and the Horizon data centres with PO's clients.

375 The key services that must be supported by these networks include the following:

- Transfer of files to and from remote systems; and
- Access to business applications and information running on remote servers.

376 Messages from branches are transmitted to both campuses and their Correspondence Servers. Messages from Correspondence Servers to branches are directed to the IP address of the Gateway PC in the branch, which is known to each Correspondence Server.

377 The ISDN card in the Gateway PC enables it to behave as a normal LAN connection. ISDN call set-up is done automatically when the Gateway PC sends a message over the link. Similarly, a call is set up (if none exists) when a Correspondence Server sends a message to a branch. To avoid too frequent calls, Riposte only sets up a call when an urgent message must be passed in either direction, or when a normal message has to be passed and a "handshake" timer expires. The timer starts at the completion of the transfer of the previous set of messages. If there are no messages to pass, no call is made, so a small branch may not catch up until the end of day. Once a call has been established then all waiting messages are transferred in both directions. The call is cleared down on expiry of an idle timer with a default setting of twenty seconds. An implication of this process is that all communications traffic between the branches and the campus must go via Riposte. There is thus usually some delay between a message being created at a counter position and that message reaching the campus.

378 With HNG, network connections are more reliable. Nevertheless, network resilience remains vital to system availability.

379 A LAN subnet is used between the two campuses. The principal need for this is where a single IP[18] address is used by client applications to access a service that runs on a server at one campus but may fail over to the other. For the single IP address to be able to follow the service, the two servers must be on the same subnet. This is implemented by bridging the two campuses to create Virtual LANs (VLANs) spanning the two campuses.

[18] Internet Protocol

Failure of Communication Link – original Horizon

380 When a branch lost its communication link or the Gateway PC, business could still be conducted. Nevertheless, counters could not communicate with the Correspondence Servers, and vice versa. Riposte message replication would not operate. The branch became progressively more out-of-date, and the campuses had no record of transactions that had taken place in that branch.

381 Certain operations that would normally make a real-time connection to the campus would time out and the user would put in a call to the help desk. When link or the PC was repaired, Riposte message replication brought the branch and the Correspondence Servers back into line.

382 At first, this was not a significant issue. The software running in branches was designed to detect failures and re-try communications until some limit was reached. The clerk may have needed to take actions, guided by the system.

383 Starting in around 2005, an increasing number of applications relied on being able to contact the campus. Banking and debit card handling were the first two such applications, but more followed. In response to this requirement, two improvements were made to the branch-campus network.

- A Counter Network Information Monitor (CNIM) was introduced. This monitored the status of the link, and if it was not available it informed the user. Once the link was re-established, this indication was removed.

- A new data network was introduced, which enabled more heavily used branches to be permanently connected to the campus.

Network Failure - HNG

384 All HNG branches detect WAN connection failures, and switch to an alternative connection type without the need for users to restart their application sessions.

385 A network failure on the LAN is much less likely because of the relative simplicity of the network. If the LAN fails, business transactions for all affected users will be prevented, so users will report the problem to the help desk who will deploy an engineer to repair or replace the faulty hardware.

386 The counter business application is aware of the state of the network by interacting with the CNIM and will not offer services to the clerk when no WAN connection is available.

8.5.6 Business Continuity and Disaster Recovery

387 Wikipedia defines and distinguishes these terms as follows: '*Disaster recovery involves a set of policies, tools and procedures to enable the recovery or continuation of vital technology infrastructure and systems following a natural or humaninduced disaster. Disaster recovery focuses on the IT or technology systems supporting critical business functions, as opposed to business continuity, which involves keeping all essential aspects of a business functioning despite significant disruptive events. Disaster recovery is therefore a subset of business continuity.*'

388 Clearly the biggest risk to continuity of Post Office's business is a disaster affecting the Horizon data centres, but the loss of a complete branch has also been considered.

389 A branch could be lost, for example as a result of fire, flood or the theft of counter systems. In the
original Horizon system, there would have been two key implications:

- The payment of benefits to customers could have been delayed. However, contingency arrangements were built into the applications that would enable benefits to be paid at any other branch. These were known as Foreign Encashments.
- Any transactions that took place within the branch, but were not replicated to the campus, would have been lost. Where the equipment, rather than the branch as a whole, was destroyed then any outstanding transactions could have been reconstituted from the paper records. Each of the printouts includes the transaction number. When the equipment was replaced, Riposte enabled the system to identify the last transaction number secured at each counter. The remainder could have been manually rekeyed.

52 HNG applications include a wider range of contingency arrangements, commensurate with the additional services provided before any disaster occurred. Because the BRDB is held centrally, the loss of a branch or counter is less problematic than in the original system. Any business being transacted at the time of the disaster is likely to be lost, but all of the previous work should have been secured.

390 DR for the data centres is discussed in sections 8.5.2 and 8.5.4 above.

8.5.7 Conclusions

391 Fujitsu has established a resilient framework to guard against disruption of Post Office's Horizon service. The framework rests on three Rs – replication, redundancy and recovery. It covers risks ranging from failures of peripherals attached to a PC in a branch up to destruction of an entire data centre.

392 The Horizon network has been steadily upgraded with alternative routes being provided.

393 Recovery is automated as far as practicable, but still depends on some user intervention under guidance.

8.6 Security and User Authentication

394 Post Office services supported by Horizon are used every working day by millions of people. The value of
their transactions represents an important part of the UK economy. It is vital that users, PO staff, independent auditors, the government and the public at large trust Horizon. This trust is underpinned by a secure computer system, qualified in broad terms as follows: '*IT security is the protection of computer systems from theft and damage to their hardware, software or information as well as from disruption or misdirection of the services they provide.*'[19]

395 This section of the report summarises Horizon security and user authentication measures, starting with the original system.

396 Access to Horizon services and system components is restricted to those who are properly authorised to

[19] Wikipedia

use those specific services and components. Authentication seeks to verify the identity of a person (or system component) seeking to gain access to a system resource.

397 Three important pillars of security are as follows:

- Confidentiality - unauthorised observation or inference of information (e.g. about benefits paid to claimants);
- Integrity - unauthorised manipulation of information (e.g. of the data passed to and from PO and its clients);
- Availability - unauthorised denial of service.

398 Vulnerabilities could cause lasting reputational damage and financial loss to both PO and Fujitsu.

399 A fourth pillar is audit and accountability: ensuring that users are held accountable for their actions by recording information about these actions. Threats are reduced if users understand that they will be held accountable for their actions. Audit is discussed in sections **Error! Reference source not found.**, 6.5 and
8.2.6 above.

400 Security and user authentication are highly technical topics. This section of the report aims to provide an introduction only.

401 As described above, the Horizon architecture includes measures to address confidentiality, integrity, availability and audit. Some of the data is supplied by the government and is classified as Restricted. Because of this, security measures must follow the guidance of CESG[20]. Other data is associated with financial transactions and so the regulations of the financial services industry are applied.

402 A detailed risk assessment was undertaken for Horizon when the system was being designed in the late 1990s, which resulted in the following security policies being adopted:

- Physical and logical access to the system is controlled, with access granted selectively and permitted only where there is a specific need. Access is restricted to people with appropriate authorisation.
- The identity claimed by a user is verified before any access is granted to the system. Authentication mechanisms also ensure that trust relationships are established between components within, and external to, Horizon.
- All users are individually accountable for their actions. Owners are assigned for all information assets. The owners are responsible for defining who is authorised to access the information. Responsibilities may be delegated, but accountability remains with the designated owner of the asset.

[20] The UK government's National Technical Authority for Information Assurance, now part of the National Cyber Security Centre

- * Audit mechanisms monitor and detect events that might threaten the security of Horizon itself or any service to which it is connected. These mechanisms also ensure that transactions and other events are reliably and securely recorded as described earlier in this report.

- * Security personnel are alerted to violations that could seriously threaten the services.

53 The main security risks that could impact branch accounts may be summarised as follows:

- * Unauthorised access – processing transactions in branch via a user account or remotely without appropriate permission; this risk includes direct access to or manipulation of branch accounts without permission. The risk is minimised by rigorous control of user identities and their access to resources.

- * Theft or damage to Horizon equipment, after which the correct position is not reinstated. Resilience to this risk depends on robust procedures for recovering from failure or loss of system components and any other dislocations of the service.

8.6.1 **Authentication**

403 Authentication seeks to verify the identity of a person (or system component) seeking to gain access to a system resource. Authority may be established using a set of credentials, most commonly a username (or ID) in combination with a password.

404 User authentication is required at specific points within the system architecture. The following types of users are identified:

a) Subpostmasters and counter clerks in branches;

b) Counter PCs that must authenticate when establishing a link to the data centres;

c) PO operations and support users;

d) PO managers, who need to access Horizon and its associated corporate systems.

405 Counter PCs are encrypted to prevent unauthorised access, particularly if they are stolen. See section 8.6.4 below for further details on encryption.

406 In the original Horizon system, the subpostmaster gained access using the Post Office Log On (POLO) process, with authentication provided by the Microsoft operating system (Windows NT at the time). This required the subpostmaster to insert their memory card into the PC keyboard and enter a PIN. The authentication process validated the PIN against information stored on the card. If that was successful, the POLO process used information from the card to decrypt the PC.

407 Login was controlled by the Riposte software, supplanting the basic Windows NT system to improve login and logout times.

408 The counter clerk submitted their user ID and password via this screen. Riposte used standard Windows NT to authenticate users, passing on the passwords to be hashed and compared with the value stored by Windows NT.

409 PCs operating in branches need to authenticate themselves when they connect to the campuses. Section 8.5.5 above summarised Horizon's networking technologies. In the original system, the gateway PC[21] in each branch periodically established a link with a campus during which all outstanding messages in either direction were exchanged. ISDN-connected branches originally used Microsoft's proprietary Remote Access Service (RAS) to authenticate themselves. However, this method was superseded by a mechanism known as VPN, which provides an encrypted channel (or tunnel) between the counter PC and the campus. Sessions are established using a VPN key distributed by a central server. The ability to initiate and respond to an encrypted connection using this key proves the identity of the branch.

410 The processes described above are used for counter PCs only. Where users needed to access the Windows NT servers or workstations located in the campuses or support centres, conventional Windows NT authentication methods were used. Similarly, servers such as those running Sun Solaris supported standard login authentication provided by Unix[22].

411 Horizon's access control policies stated that people accessing Horizon systems had to identify themselves using hand held tokens if:

- they were at sites remote from the Campuses and could update the operational systems (for example, for management purposes);
- they had access to PO business data (except at branches);
- they were authorised to update system data (such as reference data), which can affect the running of the main operational systems. This included anyone with system and database administration privileges.

412 Horizon used SecurID tokens (from Security Dynamics) as tokens. All accesses authorised in this way were audited.

413 Oracle DBMS authenticated either via the underlying operating system or directly by the Oracle database itself. Direct login to the Oracle database applications was restricted to Oracle support. Each of these had a unique user ID and password for the database (as well as separate Windows NT credentials).

8.6.2 **Roles and access control**

414 Access control is achieved by verifying that a particular user is only able to access a given resource in an approved manner. It is defined in terms of roles, each of which defines a number of functions that a user can perform. A user may be allowed to assume several separate roles.

415 Roles were initially defined as follows:

- Post Office - including manager, counter clerks and auditors;

[21] One of the PCs in each branch was designated as the gateway through which communications flowed with the rest of Horizon.

[22] Unix is the most widely used operating system not owned by Microsoft.

- * Operations - provide the means to control the Horizon systems during normal running;
- * System and Security Management - provide the means to maintain and monitor the system, including adding new software and users;
- * Support Roles - such as engineers and applications support.

416 Control of access by other computer systems is as important as control of access by people.

417 Firewalls make an important contribution to access control by protecting one part of a computer network from another. They only allow traffic to flow between a defined set of network end points on either side of the firewall using specific services. Firewalls also perform other functions such as:

a) Preventing certain users or machines from accessing certain servers;

b) Monitoring communication between networks;

c) Eavesdropping;

d) Controlling what can be sent across the firewall.

8.6.3 **Virus infection**

418 The threat of virus infection in the original Horizon system was relatively low:

- * Although the counter PCs were equipped with diskette drives, these were disabled except where they were required for transfer of encryption keys.
- * There were no e-mail connections to external systems.
- * Microsoft Word documents (which could contain Word macro virus) were rarely imported.
- * Operational files transmitted by file transfer contain only data (rather than executable code)
- * The main processing platforms were Unix based, which was less vulnerable to attack.

419 There was, nevertheless, a need to protect against the introduction of viruses from the following external sources:

- * Executable files introduced for maintenance purposes;
- * Microsoft Office documents;
- * HTML documents containing user Help information.

420 All workstations other than those in branches had virus protection software installed, which was updated regularly as new definitions and software versions were received.

421 All executable code was virus checked prior to being imported into any part of the system.

422 All Microsoft Office (and HTML) files were vetted for macro viruses before they were imported into any part of the system.

8.6.4 **Encryption**

423 Encryption is the process of encoding a message or information in such a way that only authorised parties can access it. Information is encrypted using a key generated by a special algorithm. An authorised recipient decrypts the message with the key provided by the party who sent the message.[23]

424 Horizon used encryption for three main purposes:

- to protect data on communications links that pass outside the control of Horizon, its suppliers or customers;
- to protect the integrity of individual messages from creation to use;
- to protect the confidentiality of data stored on physically insecure systems such as counter PCs.

425 A widely used encryption scheme is known as public key infrastructure. PKI uses asymmetric cryptography with pairs of keys: public keys which are shared widely, and private keys which are known only to their owners. This accomplishes two functions:

- authentication, where the public key verifies that a holder of the paired private key sent the message, and;
- encryption, where only the paired private key holder can decrypt the message encrypted with the public key.

426 In a public key encryption system, anyone can encrypt a message using the receiver's public key. That encrypted message can only be decrypted with the receiver's private key.

427 The original Horizon system used five types of encryption:

a) Symmetric encryption - for files stored on counter PCs

b) Asymmetric encryption - to seal messages for integrity, but it was not used to encrypt data

c) Digital signatures - a digital signature is a code (generated and authenticated by public key encryption as described above) which is attached to an electronically transmitted document to verify its contents and the sender's identity. A valid digital signature gives a recipient confidence that the message was created by a known sender (authentication), that the sender cannot deny having sent the message (non-repudiation), and that the message was not altered in transit (integrity). Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery or tampering.

d) One-way encryption – where the results could never be decrypted, e.g. for passwords. The original value is one-way encrypted, and the result is held within the computer system. If a user supplies a value that yields the same encrypted value, then it can be assumed that the data supplied was the

[23] This description is based on the following reference https://en.wikipedia.org/wiki/Encryption.

same as the original data. Another use for one-way encryption was to generate a seal for a piece of data. A hash value is generated which is dependent on the entire content of the data to be protected.

If the hash value is re-generated later, and found to be different to the original seal, then it can be assumed that the data has been tampered with.

e) VPNs - all traffic flowing between branches and campuses was encrypted using a variant of asymmetric encryption.

428 Horizon key management was based on the ISO 11770 model.

8.6.5 **HNG**

429 Many important principles of security were established in the original Horizon system, as described above. While HNG honours many of those, the architecture has been rationalised in that the choices are based on an updated assessment of the risks faced by the system. The security architecture has been developed with the aim of ensuring that there are no single points of failure and that each area of risk has more than one technical or management control working together to mitigate that risk.

430 The solution has been architected using the control objectives in ISO 27001[24] as a guideline.

Encryption

431 HNG still makes extensive use of cryptography and digital signatures for the protection of data, both in storage and during transit. AES[25] or TDES[26] encryption keys and RSA[27] signing/encryption keys are now used. Messages from the counter to the data centre are protected by a combination of VPN technology and SSL[28]. These transaction messages are also digitally signed.

432 Connections to third parties are protected through the use of encryption where the contractual agreement requires that.

Identity and Access Management (User Authentication)

433 The authentication of users is performed by a directory service. This includes Unix operating systems as well as Microsoft Windows. This is achieved using Active Directory as a master directory service with the implementation of a separate authentication module on non-Microsoft platforms. This enables the

[24] ISO 27001 provided requirements for an information security management system - a systematic approach to managing sensitive information so that it remains secure. It includes people, processes and IT systems and applies risk management techniques.

[25] The Advanced Encryption Standard is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology in 2001.

[26] TDES is an abbreviation of Triple DES (or Triple Data Encryption Algorithm), which is a symmetric-key block cipher that applies the DES cipher algorithm three times to each data block.

[27] RSA forms part of a PKI, providing asymmetric encryption.

[28] Secure Sockets Layer encrypts data sent via the Internet.

nonMicrosoft platforms to appear as objects in Active Directory and facilitates central access management.

434 All users of the Horizon system are individually identified, through a process controlled by the Security Team. Every administrative user uses strong two-factor authentication when logging on to the system and it is not possible to directly access any Horizon system without such a token.

Payments

435 HNG meets the requirements of the Payment Card Industry (PCI) Data Security Standard (DSS). The PCI DSS is an information security standard for organisations that handle credit cards. It was created to increase controls around cardholder data to reduce credit card fraud, and it now provides a comprehensive framework of good practice.

8.6.6 Conclusions

436 Security was recognised as a fundamental requirement of Horizon from the earliest days, rather than something that had to be 'bolted on' later.

437 Users (and other 'agents' such as counter PCs) must authenticate their identities before they are allowed to access the system. Each user, even after they have passed this first gate, is only permitted to perform one or more well-defined roles. Each role is authorised to exercise a specific set of functions and not others. This mechanism of role-based access control is established as good practice across for systems like Horizon.

438 Horizon uses encryption and digital signatures, which are also seen as industry-standard techniques, as part of its security framework.

439 We consider that the security measures employed by Horizon are at the level required for such a system.

8.7 Development and Testing of Horizon

440 This section of the report introduces the Horizon development and testing lifecycle with some of the techniques employed as well as the constraints imposed on application design and development by the system architecture.

441 A fundamental requirement of Horizon was the ability to develop new applications, and integrate them into the system, in response to new business opportunities, in a speedy and cost-effective manner.

442 Applications or their components are developed by:

- Suppliers who supply generic software such as database products;
- Third parties who supply applications or application components that meet Horizon requirements;
- Fujitsu developers who develop software to meet specific business or strategy goals.

443 The key need is that any applications from any of these sources should be capable of integration with other applications, whatever their source.

444 As discussed above, Horizon applications (branch and back office) are architected as horizontal layers. The core applications delivering Horizon functionality are supported by a series of ancillary applications - such as the Transaction Processing System (TPS), Management Information System (MIS) and a system to manage reference data. A series of further applications assist staff with system management and support services.

445 Infrastructure (in the form of hardware platforms and networks) are accommodated in the lifecycle alongside the applications.

8.7.1 Lifecycle

446 Systems development lifecycles are often depicted using a V-model and Horizon is no exception. The left side of the 'V' represents the creation of system components to meet business requirements. The right side represents testing and integration of those components culminating in validation of the solution against the original need. Horizon's model is shown below:

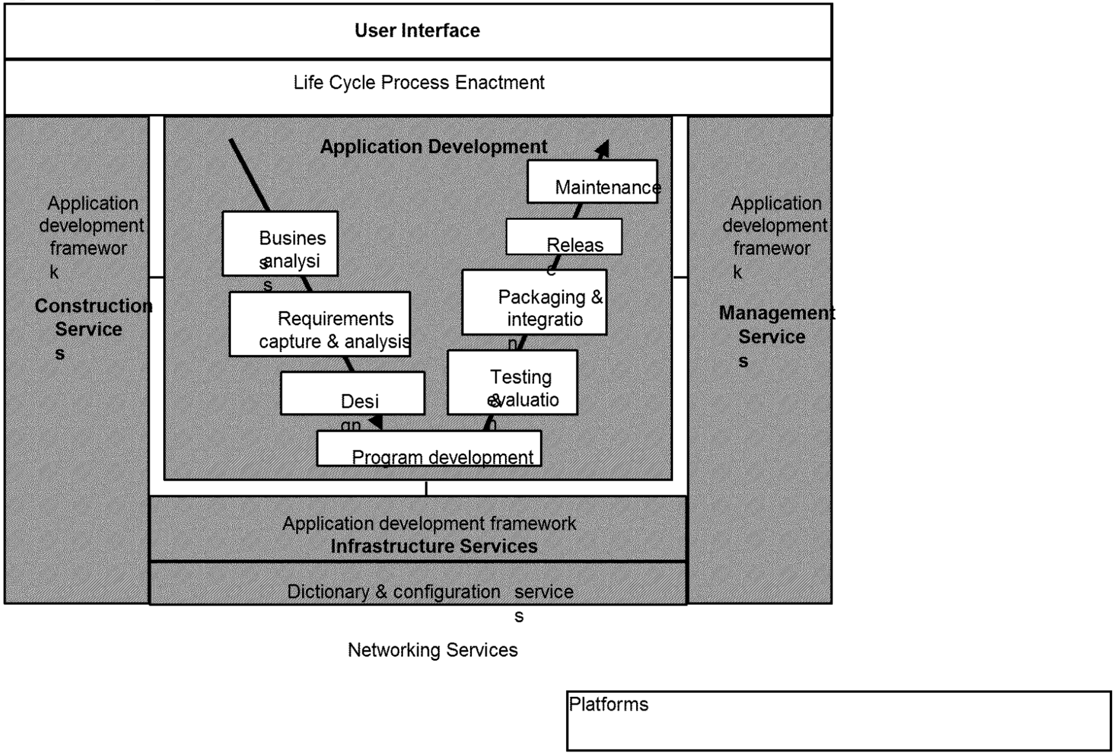


Figure 8.5 - Application development reference model

447 Most stages of the process result in documents, which are used by subsequent stages. Fujitsu's documentation is comprehensive, systematic and generally of good quality. The architecture seems complete and the system appears to be built on sound principles.

448 The application development activities shown in Figure 8.5 are further described below:

Activity	Notes
Business analysis	Initial PO requirements, new business or other changes Requirements capture, analysis and specification
Design - High Level Design - Low Level Design - Platform design - External interface design - Application design	Applications are designed using object orientation, which means that modules must be self-contained, and only communicate via pre-defined and documented interfaces that are not dependent on the application's physical implementation
Construction	Visual Basic, C and C++ programming languages were used in the original Horizon system along with Oracle development tools.
	Third party and commodity products (such as operating systems and device drivers) are simply bought in.
Testing and evaluation	Unit testing takes place once the product is developed or procured and ensures that the product conforms to its requirements. Link (or integration) testing verifies that the product interworks with other major components. This level of testing is generally performed on a complete release. Acceptance testing proves to PO's client that the development meets its functional requirements and to PO that it may be brought into use without impacting on the existing solution or other applications. We assume that Fujitsu also performs regression testing, but we have seen no evidence to that effect.
Release management	A Horizon release may be either major or an incremental set of interdependent components.

Table 8.3 - System development and testing activities

449 HNG has built on the principles of the original lifecycle but used improved methods and technology where appropriate. For instance, the move away from Riposte and Visual Basic to Java brought in more modern development tools and enabled techniques such as object orientation as described earlier in this report. The architecture now is data driven to the extent that PO can introduce new clients without reference to Fujitsu.

450 Fujitsu has explained that they follow Host Applications Design and Development Standards (HADDIS) as defined in DES/GEN/STD/0001. We assume that these form part of a more comprehensive set of standards but have not yet received evidence of that or any more information.

8.7.2 **Conclusions**

451 Horizon's development lifecycle follows best practice. The Design phase is particularly stratified with

separate consideration given to architecture, high and low-level design and interfaces. This approach gives confidence that important aspects of the design are described at all levels of detail. It has also resulted in a large set of documentation to be managed.

8.8 In-Service Support of Horizon

452 In earlier sections of this report we have discussed the rationale for Horizon, what the system comprises and how it was built. All the cases to be considered by the trial arose when the system has been in live operation supporting the business. Therefore, this section introduces how Horizon is serviced and supported.

8.8.1 ITIL

453 ITIL is the basis for the Horizon service model. Originating from the UK Government in the 1980s, ITIL (formerly an acronym for Information Technology Infrastructure Library) is a set of detailed practices for IT service management that focuses on aligning IT services with the needs of business. ITIL underpins ISO 20000, the International Service Management Standard for IT, although there are some differences between that standard and the ITIL framework.

454 ITIL describes processes, procedures, tasks, and checklists which are not organization-specific or technology-specific but can be applied by any organisation to enable its IT services to deliver the value that the business requires. The following diagram illustrates the scope of ITIL, positioning most of its key processes and important functions such as the Service Desk.

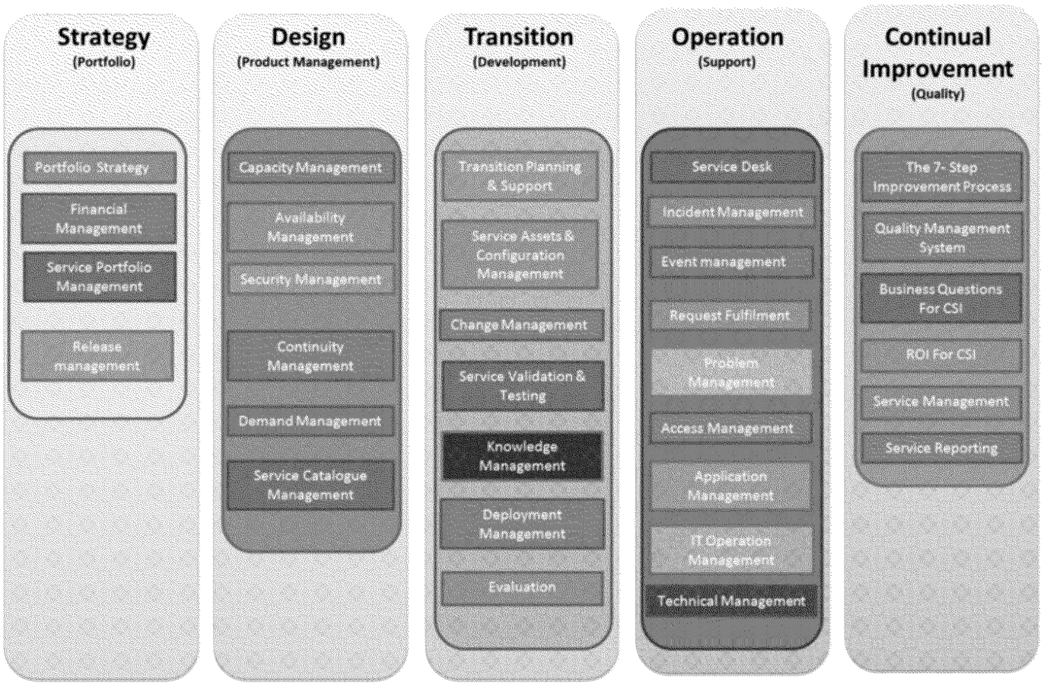


Figure 8.6 - ITIL Framework

8.8.2 Systems Management

455 The scale and complexity of the PO branch estate requires proactive and comprehensive systems

management. Every branch and individual counter position is under management and is being supported in successfully performing business transactions.

456 The same applies to applications running in the data centres. Any disruption can impact large parts of the branch estate.

457 In terms of the ITIL model illustrated above, systems management falls into the Operation column under the processes of Application and IT Operation Management.

458 Systems management facilities are needed to maximise Horizon availability and to ensure that the system delivers the service levels agreed with PO. These facilities also reduce service delivery costs as follows:

- Reducing the need for human intervention, e.g. visits to branches to install software
- Automating routine management activities, thus reducing the need for human operators
- Enabling problems to be anticipated and avoided, or their impact reduced
- Managing hardware upgrades as cost effectively as possible
- Ensuring the auditing of security events, such as authentication failures and unauthorised attempts to access resources,

459 Some of the key management products used by Horizon are as follows:

- Tivoli, supplied by IBM, was used for all services on NT platforms. It provided a central event management service, software distribution and resource monitoring facilities.
- HP OpenView together with Cisco Works are used to manage network products such as routers.
- BMC Patrol is used to handle the host central servers and the Oracle applications running on them. Patrol is specifically tailored to the management of Unix systems and applications. A Patrol Tivoli Event Adapter is provided to map Patrol events onto Tivoli events. BMC can generate pager alerts if problems arise on the platforms it manages
- Maestro, also from IBM, is used to provide scheduling facilities for batch operations.

8.8.3 **Software Distribution**

460 Horizon software runs on more than 10,000 platforms – including servers in data centres as well as all Post Office branches. Systems on this scale require some degree of automated software distribution.

461 Horizon software to be distributed to target systems is delivered (with its release notes) through a formal release management mechanism. The software is pre-packaged so that it can be delivered and optionally installed in a fully automated manner. Where such automation is not possible, the necessary manual interventions are clearly documented.

462 Reference data updates are deployed in a fully automated manner using the targeting information provided

along with the data itself.

8.8.4 **Problem Management**

463 Problem Management is an ITIL process for managing the lifecycle of *all* problems that happen in an IT service. Its primary objectives are to prevent incidents from happening and to minimise the impact of incidents that cannot be prevented.

464 'Lights out' operation at the campuses needs automated problem detection and management. Platforms generate events in response to problems. These problems are any deviations from the expected operational processing - for instance when a batch job fails, or a software error is detected.

465 Tivoli provides facilities to take events from one or more sources and use defined rules to establish whether local actions should be taken and/or whether they should be forwarded to central event servers. Event servers include functions such as the ability to correlate events, to evaluate the actions to be taken, and to schedule these actions.

466 When the automated systems are unable to handle a problem, they bring it promptly to the attention of a human operator. This person, who may be on a remote site, can manage the impact of the problem, assign it to an appropriate support team, and track processing using automated Help Desk facilities.

467 Sources of problems include applications and the operating systems as well as hardware failures.

468 All access by operations (including 2nd and 3rd line support) to manage IT systems are fully audited.

8.8.5 **Horizon support services**[29]

469 From time to time most subpostmasters and their staff have questions, problems or requests related to their usage of Horizon. PO and Fujitsu have provided a set of services and tools to support users.

470 Over the lifetime of Horizon, the organisation of support services has evolved.

471 At present, if a counter clerk or SPMR has a problem or a query with any aspect of Horizon, they phone the NBSC (Network Business Support Centre). This service is available for extended hours; it has been operated by ATOS from Manila in the Philippines since 2014. The NBSC initially aims to distinguish genuine issues from cases where the user simply needs more information or assistance. The team attempts to answer the queries or resolve the problems themselves. This stage in the support process is known as 'first line'.

472 Fujitsu maintains a Known Error Log (KEL). The KEL is primarily a knowledge base, rather than a 'bug list', which is used by the NBSC. It describes the symptoms of problems with some analysis of causes, (potential) solutions to the problems and workarounds that might be needed before a permanent solution can be implemented. The log currently comprises more than 8,000 entries. There is no defined vocabulary, which means that searching for particular terms may be unreliable.

[29] Thus far, Charteris has received limited information on this topic. Several of the documents promised by FJ at our briefing on 10 April would have been useful, but had still not been received by 31 May 2018.

473 An SMC (System Management Centre) is run by a part of Fujitsu based in India. This team monitors Horizon system operations, resolves issues internally when possible but can raise KELs (add to or amend entries in the log).

474 When either the NBSC or the SMC cannot resolve an incident and needs additional expertise or knowledge, the issue is passed to the Fujitsu SSC (Shared Service Centre) for second line support. The SSC is shared across Fujitsu customers. It is available during normal working hours only.

475 The SSC will investigate the incident, update the KEL as appropriate and invoke third line support when specialist knowledge is required.

476 The SSC runs the SSCWeb application which includes the KEL. It also enables searching of system operations events and provides access to help text as well as other support and technical information.

477 KELs may be raised by testers to flag minor issues that are not resolved immediately. In exceptional circumstances, the development group has also raised KELs to inform the support teams of potential problems coming their way. The KEL is used mainly for supporting operational users, rather than by Fujitsu's internal teams.

478 The Fujitsu MAC (Major Account Control) team manages all changes in branches. They will pass on incidents to the SSC if they need support.

479 Third line support uses a system called PEAK to log and manage incidents passed to them, which they suspect to be faults.

480 The NBSC escalates incidents via a system known as TFS[30] through a gateway into PEAK, where it is seen by the support team for the first time.

8.8.6 Conclusions

481 ITIL is widely used across the IT industry, because it provides a framework based on decades of experience in managing systems in service. Horizon system management, based on the ITIL framework, is supported by a series of market-leading tools, which enable Fujitsu to manage the full range of resources.

482 Based on what Charteris has learned so far, Horizon support services seem fragmented. They are provided

by some combination of the NBSC in Manila, the SMC in India and the SSC in the UK. Those teams are supported by PEAK and KEL and possibly other systems too.

483 Charteris is aware of indications that help services have not been consistently well received. This area may emerge as a weak link in an otherwise well managed IT service.

[30] TFS stands for TRIOLE for ServiceNow. This system is changing to ServiceNow.

9. EXAMPLE OF THE TRIPLE FILTER ANALYSIS OF BUGS

9.1 Descriptions of the Problem

484 In this section, we give an example of how we intend to use the principles and understanding of the architecture of Horizon, described in the previous section, to analyse the potential impact of specific bugs in Horizon on branch accounts.

485 The example we shall use is the 'Receipts/Payments Mismatch Issue'. This is referred to indirectly in the claim and the defence, which refer to schedule 6 of a letter from Bond Dickinson to Freeths dated 28 July 2016. The schedule states:

> *'3.1 The payments mismatch bug affected 62 branches (13 crown; 12 multiples; 37 postmasters). It related to the process of moving discrepancies into the local suspense account and majority of incidents occurred between August and October 2010.*
>
> *3.2 The identification of this bug came about through Horizon's own in-built checks and balances which are designed to flag up such issues.*
>
> *3.3 When discrepancies come to light during the rolling over of a stock unit onto a new transaction period, the user is asked if that discrepancy should be moved into the local suspense account. If the branch pressed the "cancel" icon at this stage, the discrepancy was "zeroed" on Horizon.*
>
> *3.4 The effect of this was that the back end branch account (Post Office's central accounting system) showed the discrepancy while Horizon, in the branch, did not. The branch may have thought they had balanced when they had not.*
>
> *3.5 In the affected branches, this created a "receipts and payment mismatch" equal to the value of the lost discrepancies. When the new Trading Period began, the opening figures for discrepancies in the new period was zero rather than the actual value of the discrepancy.*
>
> *3.6 The first remedial step was for Fujitsu to ascertain which branches were affected. The mismatch generated an error code which allowed Fujitsu to identify the relevant branches. Fujitsu were then able to carry out analysis on each affected branch to gather relevant information. For example, they needed to ascertain when the receipts/payments mismatch occurred, the value of the lost discrepancy and whether it was a gain or a loss.*
>
> *3.7 There were 17 postmasters who had a loss attributed to their branch. They were notified of this in March 2011 and, where appropriate, they were reimbursed. Postmasters who made a gain through the anomaly were not asked to refund this amount to Post Office.'*

486 This description is a summary from two notes produced at the time, 'Receipts/Payments Mismatch Issue Notes (a summary of a meeting involving PO and Fujitsu staff), and "Correcting Accounts for 'Lost' Discrepancies" (a note by Gareth Jenkins on 29 September 2010).

487 There is also a more detailed analysis of the error, in a document 'Receipts&Paymentsv0.4.docx' from Gareth Jenkins dated around May 2011, in which he provided a sequence of screenshots and sample reports obtained by reproducing the fault in a test system. In order to understand the nature of the fault,

we have cross-referenced this set of screenshots with the process sequence for balancing a stock unit (use case BAC-11, at page 20) in the high-level design for branch accounting at the counter, DES/APP/HLD/0126. This has enabled us, following a verbal description of the issue by Gareth Jenkins, to understand the origin of the fault in some detail, although we will not describe all that detail here.

488 While the extract above is an appropriate summary of the contemporary notes, paragraph 3.3 does not make clear all the conditions required for the error to occur. In order for the error to occur, not only did the user have to press 'cancel', but they would next have to re-attempt to roll over to the next balancing period - rather than cancel the rollover. So, the condition is doubly rare, which is why it had affected so few branches.

9.2 The Branch Balancing Process

489 In what follows, it is worth noting that:

- A Trading Period (TP), contains either four or five Balance Periods (BP), which are a week each.
- A Branch contains a number of stock units (SU).

490 We are concerned here with rolling over a branch to a new TP. To allow the SPMR to do this, it is necessary that every stock unit in the branch should be in balance - for instance, that the cash recorded in Horizon for the stock unit should match with the cash physically counted. During the TP, the stock in the stock unit may or may not have been balanced at some BP (during the Trading Period); if not, it must have been balanced at the start of the TP (to allow the previous rollover).

491 In HNG, no transaction data or stock quantities are held permanently in the branch software or hardware. At the start of the branch balancing process, all the necessary information has to be retrieved from the BRDB to the branch and stored there in a cache (in volatile memory). The cache is used for computation and for displaying information to the user, and user input alters information in the cache. But any of those computations or user inputs will have no lasting effect until information is returned from the cache to the BRDB over the network.

492 We first describe some of the measures for intrinsic error prevention.

493 HNG is designed so that any failure of hardware or the network, at any stage in the branch balancing process, should not leave the BRDB in an inconsistent state, and should allow the process of branch balancing to be continued from a consistent state after the hardware or network is recovered. In this sense, a failure at any stage of branch balancing, or even a user decision to cancel some operation, must be catered for. This is normally done by making all updates to the BRDB in success units (which either

completely succeed, or completely fail with no effect on the BRDB); and by making those success units be zero-sum baskets of accounting postings wherever possible.

494 We shall illustrate this approach by the balancing of a branch with several stock units. If more than one SU is out of balance, it should not be necessary for the SPMR to provide cash to balance each one individually; this would just cause extra work, compared with providing or withdrawing one cash sum to balance all of them. This is achieved in HNG using local suspense accounts. As the SPMR goes through balancing each SU, if he finds a discrepancy, he does not need to sort it out immediately; he can park it in a local suspense account, and then balance that suspense account once as a whole after all SUs have been checked. (Here 'local' means 'specific to the branch'; it does not mean 'stored in branch hardware', as all accounting data are stored in the BRDB.)

495 When a discrepancy is discovered in one SU, the software at the branch makes a zero-sum basket to move the discrepancy to the local suspense account for the branch, recording that in the BRDB. This basket will either succeed entirely or fail entirely. Thus, in the event of hardware or network failure at any time, when some but not all SUs have been balanced, it will be possible later for HNG to discover from the BRDB which SUs have not yet been balanced and allow the user to take up the process where he left off. This is because, at all stages in the process, the local suspense account can be assumed to record discrepancies for all SUs whose balance has been tested.

496 The final process of balancing the local suspense account, by adding or withdrawing cash, is also recorded as a zero-sum basket. The user enters cash and initiates the success unit. He is informed whether it has succeeded or failed; if it fails, having put cash in the drawer, he can retry it.

497 The process is recorded in the Branch Trading Manual:

> *'When each stock unit is rolled over to the next Branch Trading Period the value of any unresolved discrepancy is transferred to Local Suspense. The last stock unit to balance must clear Local Suspense to a zero balance otherwise the Horizon Online™ system will not allow the balance and rollover into the next Branch Trading Period to continue.'*

498 We next illustrate the process of correction of user errors. Suppose that the SPMR either inaccurately counts the cash in one SU, or inaccurately counts the cash he supplies to balance the local suspense account. In these cases, Horizon will think the branch is in balance; and the amounts of cash recorded in Horizon will not match the physical cash in the branch. If there is missing physical cash in the branch, then at the next branch balancing process there will be an imbalance of that amount; the SPMR will have to supply cash to balance, but to do so one TP or one BP later than he would have done had he counted accurately in the first place. So, these user errors delay the process of balancing, but do not ultimately lead to incorrect balances.

499 We next look at the process of balancing one SU. In order for Horizon to produce its own estimate of the cash in a stock unit, for comparison with the physical count of the stock unit, Horizon needs to do three things:

a) Find out the physical cash in the stock unit when it was last balanced or recorded. This depends on two pieces of information in the BRDB: (1) the Horizon recorded cash in the stock unit at that time; and (2) a record in the BRDB of whether the stock unit was in balance at that time - or if it was out of balance, by how much. Horizon adds these two amounts. This addition is done by software in the branch, during the balancing process.

b) Find out the changes in stock or cash since the stock unit was last balanced. This involves finding all changes in the stock unit since the last balance and accumulating them so they can be added to the amount of stock at the start of the period. This calculation is done in the back end, close to the BRDB, to avoid having to send all change records for each SU back out to every branch. In practice, Horizon re-computes the change in the amount of stock in each SU at the end of each day to avoid having to add all the changes during a BP or TP for all branches at the same time each month - the end of the TP. Having recomputed stock daily, for the TP, Horizon only needs to add up changes during the last day.

c) Add the physical stock at last balance (from (a)) to the changes in stock since last balance (from (b)) to obtain Horizon's estimate of physical stock in the stock unit, for comparison with the stock count. This addition is done by software in the branch, during the balancing process.

500 If all communication from the branch to the BRDB is done in terms of zero-sum baskets, this provides a powerful check against errors in the branch software which might affect branch accounts. Any such error would put the branch cache in a state where some financial figure was incorrect. However, this figure could only affect branch accounts by being sent to the BRDB in a zero-sum basket. If one figure in the basket is wrong, it is very unlikely for other figures in the same basket also to be wrong, in such a way as to preserve the zero sum of the basket. If the basket is not zero sum, it will be rejected by the BAL, or by the software in the branch which sends the message to the BAL; so, these software errors would be detected immediately and would need to be corrected.

501 However, it is not possible in all cases for the communication from the branch to the BRDB to consist only of zero-sum baskets. At some stage in the branch balancing process, it is necessary for the branch software to tell the BRDB that 'this SU is now in balance'. That information is stored in a table in the BRDB as a single entry saying, 'at date X, the imbalance in SU Y was zero'. This is not a double entry, so there is no double entry, zero sum constraint to check on the update to the BRDB. This is how the error occurred.

9.3 How the Error Occurred

502 The error occurred through the following sequence of events:

* At a Balance Period in the middle of a TP, a user erroneously believed he was going on to do a TP rollover (which would only be allowed at the end of a TP).

* He balanced a number of SUs, finding an imbalance in the one of them.

- When asked if he wanted to move the imbalance to a local suspense account, the user realised his mistake and cancelled.

- Had he completely cancelled the balancing at this stage, no harm would have been done, because nothing more would have been sent to the BRDB. However, he chose to complete the balancing for the BP.

- This put the cache into an inconsistent state, which did not record the imbalance of the SU.

- The branch software then informed the BRDB that the SU was in balance. Because this operation was not a zero-sum basket, the inconsistency in the cache was not detected.

- The branch software informed the user (erroneously) that he was in balance. The user did not spot the error, or if he did, did not act on it. For instance, some users could have withdrawn cash, and did not do so.

- The zero figure in the BRDB for the imbalance in the SU was inconsistent with the trail of accounting entries against that SU in the BRDB - but there was no regular check for this inconsistency.

- At the next balancing of the SU, the erroneous zero balance from the BRDB was used as the starting point. This allowed the user to balance the SU by adding or withdrawing an incorrect sum of money.

503 Therefore, the error occurred only when the user took a very uncommon path through the process (erroneously starting to balance for a TP in the middle of the TP; spotting his mistake and cancelling, then continuing with a BP balancing).

504 It was not detected by the software because it involved an uncommon path, and that path involved a comparatively rare operation, of updating the BRDB outside a zero-sum basket, and so not making a zero-sum check.

505 However, the error did not go undetected. We have described in section 8.2 how in large IT systems, information is stored redundantly and there are many consistency checks of the redundant information. Fujitsu detected the error through system events, as described in Gareth Jenkins' first note:

> *The receipts and payments mismatch will result in an NT event being generated. These use event id of 902 when detected during SU balancing and 903 when detected during BTS production.*
>
> *Also application event 116 or 117 should be written to the BRDB_TX_REP_EVENT_DATA. Therefore an extract from BRSS of all instance of events 116 and 117 will provide a further check.*

506 For our expert report we intend to analyse the causes of these events, to see how they are caused, and whether they are caused by consistency checks of the type we have outlined here.

507 It is noteworthy that two independent checks both detected the error.

508 These checks enabled Fujitsu to identify the bug within about 2 months of its first occurring, and to

identify all the affected branches. All those who had lost money through the error were compensated.

9.4 Triple Filter Analysis, and Significance of the Error

509 We intend to analyse all potential bugs in Horizon using a triple filter, which consists of:

a) Would the bug have any effects on branch accounts?

b) Would the measures for detection and correction of user errors have cancelled the effects of the bug?

c) Would the measures for intrinsic error detection have cancelled the effects of the bug?

510 The last two parts of the filter have been described in sections 8.1 and 8.2.

511 In terms of the triple filter analysis, the analysis of this error is:

a) The bug does affect branch accounts.

b) The bug is caused by the user taking a rare and unlikely path through the branch balancing process, rather than actually making an error, so the measures for user error detection did not detect it.

c) In this case, the error was not caught immediately by intrinsic error detection measures, because of the rare circumstance of updating the BRDB outside a double entry transaction. Therefore, it did affect a few branch accounts. It also produced misleading reports at the branches. However, other consistency checks in Horizon did detect the error later, in two independent ways, and all adverse effects on branch accounts were cancelled.

512 It should also be noted that the error occurred in September 2010, shortly after the introduction of HNG. Introduction of new software is the time when errors not detected in testing are most likely to occur; if these errors are corrected promptly, the rate of new errors is not expected to stay so high in ensuing months.

513 To summarise the history and impact of this error:

* It was only triggered by users following an uncommon sequence of actions.

* It passed the first two tests of the triple filter.

* It was detected the third filter of intrinsic error detection, but not before it had had some adverse impact on branch accounts.

* It was detected by two independent checks in Horizon.

* Only a small number of branches were affected (60 out of 11,000).

* Branches were only affected over a short period (2-3 months).

* The monetary amounts were small in all except a few cases. The PO's contemporary analysis describes 'an overall cash value of circa £20K loss'

- * It occurred following a new release of the software, when the incidence of new errors is expected to be higher.

514 Therefore, the history of this error tends to reinforce the view that Horizon is a robust system, rather than cast it in any doubt.

APPENDIX A GLOSSARY OF TERMS

515 The terms used in this report are defined in the table below:

Term	Meaning
APOP	Automated Payment Out-pay Database
APS	Automated Payment Service
DBMS	Database Management System
DRS	Data Reconciliation Service
DWH	Data Warehouse
LFS	Logistics Feeder Service
POS	Point of Sale
RDDS	Reference Data Delivery Service
RDMC	Reference Data Management Centre
TES	Transaction Enquiry Service
TPS	Transaction Processing Service
VPN	A Virtual Private Network is a secure channel that appears to be private despite being carried on a public network, typically the Internet.
WSPOS	Web Services POS