Implementing A Resource Chargeout System

Paul Shipley
Pleasal Enterprises Pty Ltd

This paper describes some of the issues to be considered in taking chargeout rates and policies, and implementing them for a resource based chargeout system.

1.INTRODUCTION

Resource based chargeout is charging the clients for the amount of use they make of computer resources. This is similar to use of the telephone, where the final cost is based on the use made and the type of product. In the case of the telephone it depends on whether the call is local, STD, ISD, Operator Assisted, Freecall 008 or Recorded Information 0055. Each of these products have different charging methods and rates. In the case of computer chargeout, the final cost depends on usage of CPU time, disk space, tape rental, lines printed and various other measures.

Having decided to implement a resource based chargeout system, the things to be considered are:

The products to be charged for and their rates.

The impact the chargeout system will have on resource usage.

The chargeout software to be used.

The means of identifying users.

Reporting and client inquiries.

2. WHAT SHOULD BE BILLED?

You obviously have to choose measures that are accurate and apportional to be your products. The problem is that there is such a wide choice of products to be billed in the average sized IS department or data centre. There is: CPU time, media mounts, real memory usage, I/O's, disk space, tape rental. Also databases such as DB/2, ADABAS, IDMS, IMS. There may be some more local products like labour hours, PABX, data networks, PC software, help desk. You need to know which of these to bill and which to ignore, because billing some of these can be extremely unproductive. Table 1 describes some of the measures that can be used [MERR84].

In the past it was common to bill for real memory use, however this meant that jobs using significant virtual memory cost less during periods of real memory constraint. Therefore clients would submit these jobs during peak hours because it was cheaper to do so. Just what you needed! Since the amount of real memory used by a job is beyond the control of the client, why bill for real memory? It is better to include the cost of the memory as part of the overall CPU time rate calculations and ignore the memory usage.

3.CHARGEOUT IMPACT

Whatever you choose to bill will have an effect on the habits of your clients. It is very important to know not only what you bill, but what you do not bill and why.

3.1.Billed products

The introduction of a chargeout application will have an impact on the way the clients use the system. The extent of this will be determined by the importance placed on the financial results. Are the charges just "funny money" off a budget or is it off a bottom line. Either way what you charge for, will have an effect on client behaviour.

Charging for TSO logons will cause clients to want to logon as few times as possible. This will possibly lead to many clients logged on with very little activity.

Conversely, charging for TSO connect time will cause clients to remain connected for as short a time as possible, causing many logon / logoff sequences for each client.

Unless you are deliberately trying to change client behaviour it is better to leave such complexity alone.

3.2.Unbilled products

If you choose not to bill for a product either due to a staged implementation or problems which prevent its billing proceeding, it is important to monitor its usage to ensure that the particular product is not being overused.

In an actual case, the charging of a widely used product was delayed due to difficulties in implementation. When it was finally implemented it was found that the biggest use of the product was by the application ranked sixth in overall size, accounting for 30% of charges for the product. The overall largest application was the next highest with 5% of charges for the product. All other applications used much less than this. There was no apparent reason for this unexpected misalignment other than the product was not being charged for.

3.3.Discounts and Off-Peak charging

Most data centres will have times during the day, week or month when there is spare capacity. The chargeout system can be used to prompt clients to move some of their work into these times. However don't be over generous or you may find that your peak hour moves into the discount period, with a significant loss of revenue.

4.CHARGEOUT SOFTWARE

There are a number of packages on the market which measure computer resource usage, some include chargeout modules. All of these packages work for someone, the important factor is what works for you. It is critical to get the right package at the start since starting over can be very expensive!

All of the available packages seem to measure usage, the problems arise with their flexibility and the financial interfaces.

4.1.Flexibility

Any chargeout system must be flexible in order to accommodate new products and the changing environment.

Once you start billing, inevitably you will be required to bill for new products, including some you may have not considered, such as: development team labour hours, network infrastructure, PABX. If your package can not handle these products you could have major problems.

The package should also be evaluated for future growth in both volume and products.

If your package requires you to have all of the data on one disk volume and this year's data only just fits, you should not be surprised when next year's doesn't. This seems obvious, but...

4.2.Financial Interfaces

In most companies the chargeout package will not be used in isolation, but will be interfaced to other applications, such as the accounts receivable or general ledger. Also other charges need to be integrated for invoicing. This is the area where packages should be evaluated carefully, since it is pointless to buy a package only to have to rewrite it to suit the local environment.

The simplest approach in concept is to take your existing resource data from your capacity monitoring application and interface it to your existing accounts receivable / general ledger by applying rates, etc. At the other extreme are fully integrated packages which will take raw usage data (Eg: SMF) and process it into your other applications. The correct approach for you will depend on the skills available to your organisation.

5.CLIENT IDENTIFICATION

One of the most important decisions to be made is the means of identifying your clients.

In general, if you can bill an entire product or application to one client, then it is far simpler to chargeout that way. However your client may want to measure their charges against individuals or work groups so you may still have to go to the effort of further identifying clients.

5.1.Types of Identifiers

Depending on the product, varying identifiers are available. Batch Jobs can be identified by job name, account code, RACF group and userid. CICS can be identified by account code, userid and terminal. Table 2 describes some of the identifiers that can be used [MERR84].

5.2.Overheads

Some of the charges incurred will be by various support groups within the Information Technology organization. For example, who pays for the system resident volume, system programmer's logons, and even the chargeout application? There are several ways of handling this:

Suppress these charges from the chargeout application completely. The problem with this is that you cannot measure the cost of support groups.

Separate the overhead charges after determining the cost, but before posting to the financial applications.

Process them as with any other charge.

There are probably other ways, how you handle this will depend on the policies of your financial areas.

6.REPORTING AND ANSWERING CLIENT INQUIRIES

It seems inevitable that as soon as you have sent out an invoice to a client, they will be on the phone querying it. It is your responsibility to provide the answers.

6.1.What data to hold and for how long

The chargeout application will generate massive amounts of data, the problem is to know which of it should be kept and for how long.

At one end is raw data such as SMF. The problem with keeping this data for any period of time, apart from the volume, is you must have the software that you originally used to process it if you want to reproduce your original results. The latest software version may be able to process the old data but the results may be different. It is probably better to keep the processed results, such as the cost of each job with its other details than to try and reproduce these from the SMF data.

At the other end is the invoicing data, which may have been sent to the client or posted to the general ledger. This data should be kept for at least the same period as the general ledger data.

Intermediate data would have to be considered individually depending on how useful it is in helping to answer users queries.

6.2.Reporting

Your clients will quickly overwhelm you with requests for information and reports, regardless of how many resources you commit to the task. The best solution to this is to have a collection of standard reports which are easily executed with parameters (eg: via ISPF).

For enquiries that can not be solved by these reports, have a mechanism to provide the client with the appropriate data, in a form that they can manipulate themselves. This may take the form of a spreadsheet, if that is what your clients are most comfortable with.

For larger organisations, an enquiry application for clients to query their own charges may be necessary.

6.3.Analysing source data problems and corruptions

Often (hopefully not too often) your clients will enquire about data problems, usually overcharging. A fair proportion will be due to simple misunderstandings of the chargeout mechanism.

The remainder will require tracing the charges from the financial data back to the source. Usually the problem will stand out from the others in some way. By following the exceptional charges back into the more detailed data you should find the source of the problem.

In a real case, a client complained that his charges had risen dramatically this month. Comparing this month's summary with the previous month confirmed the client's problem and revealed that the source was Bull Timeshare CPU. The daily summary showed that one day was over a hundred times higher than normal. The detailed logon data indicated that this was entirely due to one logon. When queried the client remembered a day where his terminal "locked up" and he switched it off and went home. His session had been in some sort of loop and was not cleared until timeshare was shutdown overnight.

6.4.Abends, Reruns and other disputes

One area of common dispute is who pays for jobs that abend and have to be rerun. These situations are best handled on a case by case basis.

Your clients might try to convince you that they should not have to pay in the event of certain abends (because they are caused by operational errors). This should be resisted. While there may be the best of intentions on both sides, it is very easy to deliberately cause an abend. The MVS ABEND macro can be coded to generate any user or system abend [SUPER87].

6.5.Why my $1.00 job cost $3.50

This sort of problem is usually caused by two very difficult, yet essential, concepts: Capture Ratios and System Multipliers.

6.5.1.CAPTURE RATIOS:-

are used to match one set of data to another. In the current context they are most often used to match the CPU time measured by SMF with the actual CPU time used as recorded by RMF. While the means of determining capture ratios is beyond the scope of this paper, they can vary between types of loads and software releases. As such they should be reviewed occasionally (Eg: every six months or when major new software releases occur).

Capture Ratios can be a very difficult concept to explain to clients who are neither computer or statistically literate. Indeed even application programmers can find this subject to be heavy going.

6.5.2.SYSTEM MULTIPLIERS:-

are used to match the amount of CPU time measured on one processor with the measured time of a processor of a different speed. For example, a processor with twice the speed should take half the CPU time to run a given job. If you are billing by CPU time then the job would cost half as much on the faster processor. This may be welcome, but could result in clients wanting to run all of their work on the faster processor, leaving you with a under-utilized slow processor.

A solution to this is to include a system multiplier of two into the algorithm for the faster processor. This has the effect of standardizing the costs of jobs between the two processors. The problem with this approach is that if the client takes the raw CPU time (as reported on the job log) and multiplies it by your advertised rate, the cost they arrive at will be significantly different to yours.

An alternative solution is to have different rates for the two processors. However since the rate for the faster processor would be twice that of the other processor, the clients will not want to use the faster processor since it appears to be expensive. Incorrect, but obvious.

7.AVOIDING OPERATIONAL PROBLEMS

The chargeout application should be treated in a similar manner to other production applications. If you treat it as an inconvenience that distracts from your "real" work and that "near enough is good enough", then you can hardly be surprised when your clients do not trust your results.

There are three main operational problems: Duplicate data, Missing data and Corrupt data. The first two should be addressed by your chargeout package, otherwise you will have to build your own tools to prevent / monitor them. While investigating corrupt data was touched on earlier, it is better for you to find and correct the problem, rather than your clients telling you. If your package does not have some sort of profiling or trend analysis, then you will have to build one.

8.FUNCTIONAL BILLING

Functional Billing is where users are charged in units they understand, such as accounts processed, invoices printed, etc. Since these units relate to business function this type of billing is called Functional or Transaction Billing. While there is a wide interest in this, it is nearly impossible to achieve in practice and is of questionable use.

8.1.Implementing Functional Billing

The first problem you encounter is obtaining the business function based usage data, since all of the currently available measures (ie: SMF) are based on system units. A number of people have tried matching system level transactions to business functions, with varying degrees of success. However this matching will be invalidated with the next software release. While your change control procedures can be modified to ensure that new releases are not implemented until changes have been investigated for their impact on functional billing, this is committing you to a process of continual investigation. Finally, the reliability of this matching may be uncertain.

The only reliable way to obtain business transactions is from the application itself. If the application is already logging this type of data then you are in luck, but make sure that it includes enquiry transactions, otherwise you could be in for a surprise! If the application was developed in-house you may be able to get the developers to add transaction logging. Otherwise if the vendor is not interested in adding this functionality your options are limited. You are still committed to continual updates, however this process is much simpler than matching system transactions.

8.2.Problems with Functional Billing

Apart from the above problems, there are many others.

The rates (as seen by the clients) are no longer determined by the cost of CPU seconds but by the efficiency and complexity of the application code. If the application developers double the CPU usage of a transaction, you will have do the same to the cost of the transaction. Unless you can make the developers accountable, it will be very difficult to convince the clients that you are not responsible for this.

While you may be charging a set rate for a certain type of transaction, the CPU time taken for the transaction will vary depending on various system factors, such as load and database disorganisation. Who carries the business risk in funding the difference?

When there are disputes over charges, who is going to be responsible for their investigation? If the developers have changed something and it affects the billing, how will you know? As in the previous case, unless the developers can be made accountable, you could be faced with having to trace their programs for the answers.

Of course the developers are primarily there to create and modify the functionality of the applications. They have their own deadlines and schedules and are quite often being driven in different directions to you by the same clients.

8.3.Some solutions

The problems with Functional Billing are predominantly political rather than technical. Functional Billing was a response to clients legitimate desire to determine the costs of their business. For example: How much does it cost to open a new account? While clients may ideally like to charge a set fee for each new client transaction, for the reasons I have outlined this can be very difficult and expensive to achieve. However this does not mean that the question cannot be answered.

Usually it is quite easy to determine overall totals for the required information. For example, the total number of new accounts on a certain day and the total cost of the new account transactions could be determined. From these the average cost can be determined. The developers should be able to generate statistics by client area (eg: New accounts per branch) which the clients can use to determine their costs. This also has the advantage that if the average cost rises and your rates have not, then it must be the developers fault!

9.CONCLUSION

The implementation of a chargeout application is a combination of technical and political challenges. As such, several skills other than the straight technical ones, will be required to win the trust of all the parties involved. If you believe that buying a package is the end of the problem, then yours are just beginning.

REFERENCES

[MASON92] Mason, Phil, Chargeout Methodologies, CMGA Journal, November 1992, Turramurra, NSW

[MERR84] Merrill, H.W."Barry", Merrill's Expanded Guide to Computer Performance Evaluation Using the SAS System. SAS Institute Inc., Cary, North Carolina, USA

[SUPER87] International Business Machines Corp., MVS/XA: Supervisor Services and Macro Instructions, Poughkeepsie, New York, USA, GC28-1154-3

ABOUT THE AUTHOR

I have worked in the computer industry since 1984, mainly developing applications in Cobol, SAS and Focus. I have spent the past three years as Technical Leader for Taconet Billing, the Telecom chargeout application.

TABLES

Product	Measures
Batch	CPU Time Job Class Start / End Time Tape Mounts (Private, Scratch) I/Os Elapsed Time
TSO	CPU Time Logons Logon / Logoff Time Connect Time I/Os
CICS	CPU Time I/Os
Disk	Mega Byte Days Type of device Creates / Deletes
Tape	Tape Days Type of media Create / Deletes Silo / Library / Offsite
Printing	Lines / Pages printed Form type Data centre / Client

Table 1: Measures that can be used for chargeout

Product	Identifiers
Batch	Account code RACF Group Userid Job Name
TSO	Account code RACF Group Userid Terminal
CICS	Account code (User extension) Userid Terminal Application ID Transaction name
Disk	High Level Qualifier Volume Serial
Tape	Account code Userid High Level Qualifier Volume Serial
Printing	Account code (when matched to Job records) Job name Output device name

Table 2: Identifiers that can be used for chargeout