Developing Cash Flow Forecasting for Service Contracts

William S. Calvert, Service Research, Inc., Rockville, Md.
Margaret K. Calvert, Service Research, Inc., Rockville, Md.

ABSTRACT:

Development of financial forecasting applications for service contracts is discussed with particular attention to the use of modular algorithms, validation, and table driven architecture. The general approach for developing and implementing the cash flow forecasting algorithms presented is applicable to a wide range of contracts or long term service arrangements where the income generated depends on the delivery of units of service. Two basic algorithms are presented and discussed in the context of validation and performance concerns. The prototype application discussed was developed under OS/2 but the code and principles presented are generally applicable to other platforms.

INTRODUCTION:

In these times, the market place is very competitive. High quality financial information has been recognized as essential to form the sound strategic judgements that lead to winning decisions. (Fingerman, 1988). Knowledge of cash flow is part of the financial information required to operate a business effectively. Often, however, where business involves complex and/or long term contracts and business interactions, cash flow can be difficult to estimate or predict. (Calvert, 1993; See also Gilliland and LaBella, 1993 for a common sense approach to complex forecasting problems).

In the case considered here, units of service are provided on an "as needed" basis, limited by total cost or the end of the contract term, whichever occurs first. The "need" may follow a pattern or occur almost as a random process. An application system was developed to systematically calculate the potential cash flow generated by historic, new, and anticipated contracts for various business segments and a range of conditions for this kind of business contract. The resulting reports contain both projections and forecasts, but will generally be referred to as a forecast in our discussions.

Forecasting cash flows in the manner described may be applieable to equipment maintenance services, clinical testing services, or some technical support and consulting contracts. The algorithms presented here are the key ingredient to a forecasting application of this sort but represent only a very small portion of the code that must be written to provide the user interface, data acquisition and validation, security and backup of data, as well as the preparation of reports and visual displays of the newly generated, current actual, and historical data. (See Fingerman, 1989 for more discussion on how a system may be produced.)

DEVELOPING THE APPLICATION:

The application was developed working closely with client staff using a rapid prototyping method. The client/user had specific requirements and strong insights into the overall requirements of the system, however, there were some conceptual and communication difficulties due to inexperience with computing and information systems. The prototyping method of development worked well in this circumstance, though the repeated rework of the system sometimes was trying on the patience of all.

Much of the rework and 'fixing' of the application system was eased by the overall design that evolved after the first couple of major revisions to the specifications. Since the exact operating parameters seemed to change frequently and over a wide range, a table driven architecture was developed so the user could select a range of options and parameters and produce a very extensive range of forecast reports and graphics. In the later stages of development, often only the tables needed to be updated to reflect changes in specifications or requirements.

Significant problems were encountered with data quality. Since the data was acquired in several machine-readable forms as well as hardcopy forms, normal data entry controls that could be implemented were limited. The choice was made not to resolve the data problems at the source, but to try to use the data "as is". The data is reasonably complex, due to the complexity of legacy business processes. No completely satisfactory screening of the data has yet been attained. The data screening in use provides protection for the sensitive logic of most of the code, at the expense of eliminating questionable records. Some money is not accounted for! The application can never achieve reliability until the data problems are resolved.

HOW THE APPLICATION WORKS:

A simplified view of the application is presented schematically in Display 1. Note that the main sources of data are external to the forecast system. Only part of the data is maintained by the application user. (This is a major weakness that we may be able to change before publication.)

Since the application may be entered at any point on subsequent occasions, there is some redundancy in processing. Global variables and parameters are stored in a table (SAS® data set) and moved into macro variables for each task. This insures that the latest system state is always the one that is used.

Note that a different algorithm could be used for any one series of forecasts. Structurally, each record could use a different algorithm, providing the user wished to designate a specific algorithm for a specific account. Implementation of this feature was delayed due to the difficulty of identifying the best algorithm for each contract.

The reliability and validity of the algorithms is important to the system because they perform the calculations that are the basis for each of the three streams of information that are produced. Several algorithms were developed using a modular structure to minimize the problems with maintenance and validation.

DEVELOPING VALID ALGORITHMS:

The first algorithm (presented in Display 2) is simply a tabulated rate of spending. At any given point in time a contract is expected to produce a proportion of its total value. Safeguards are built in to prevent over spending. Excess value is spent at the end of the contract. This may happen if spending is slow in the earlier time periods of the contract.

There are three components of validity to be considered for any algorithm: One, the correctness of the code and the system in which it is implemented. Two, the correctness of the values in the table. And three, the appropriateness of the assumptions about the process being modeled.

The code was first validated in a test system. A variety of 'expected' data was fed into the algorithm. After it was processed, the output was examined closely. A number of records were carefully traced through the process. Any anomalies were investigated and problems were resolved as necessary. In the second stage of code validation, the algorithm was put in the complete system and fed the same test data. After getting the same results as before, the algorithm was assumed to be properly installed in the system. The final stage was to use real data and look closely for any unexpected results. Any problems were traced and resolved as necessary.

The tabular values were determined from historical performance of similar contracts. The data was smoothed by eye and in some cases modifications were made 'authoritatively' to assure that the projections would be conservative (that is err on the low side). Subsequent to producing the original table, forecast results computed using the table were compared to forecasts generated by other means. After a refined examination of historical data the table was adjusted to produce acceptable results.

The assumptions associated with the use of this projection algorithm cannot be directly validated against the process that is modeled. Predictions have a way of being self-fulfilling. Retrospective comparisons are used to estimate reliability, however, there are often things that change and must be accounted for in making a retrospective comparison. Face validity provides the strongest evidence for the ultimate usefulness of the algorithm. That is, it is reasonable to make the assumption that new work will be performed in much the same way as old work, and that on average new contracts will perform in much the same way as old contracts. Perhaps the chance occurrences that influence the behavior of contracts prevent forecasting to a greater level of precision.

The second algorithm (presented in Display 3) is simply an average daily rate of spending. At any given point in time the remaining value is expected to be distributed uniformly over the remaining days when service is to be provided. As with the previously discussed algorithm, safeguards are built in to prevent over-spending and any excess value is spent at the end of the contract. This is usually just a small adjustment.

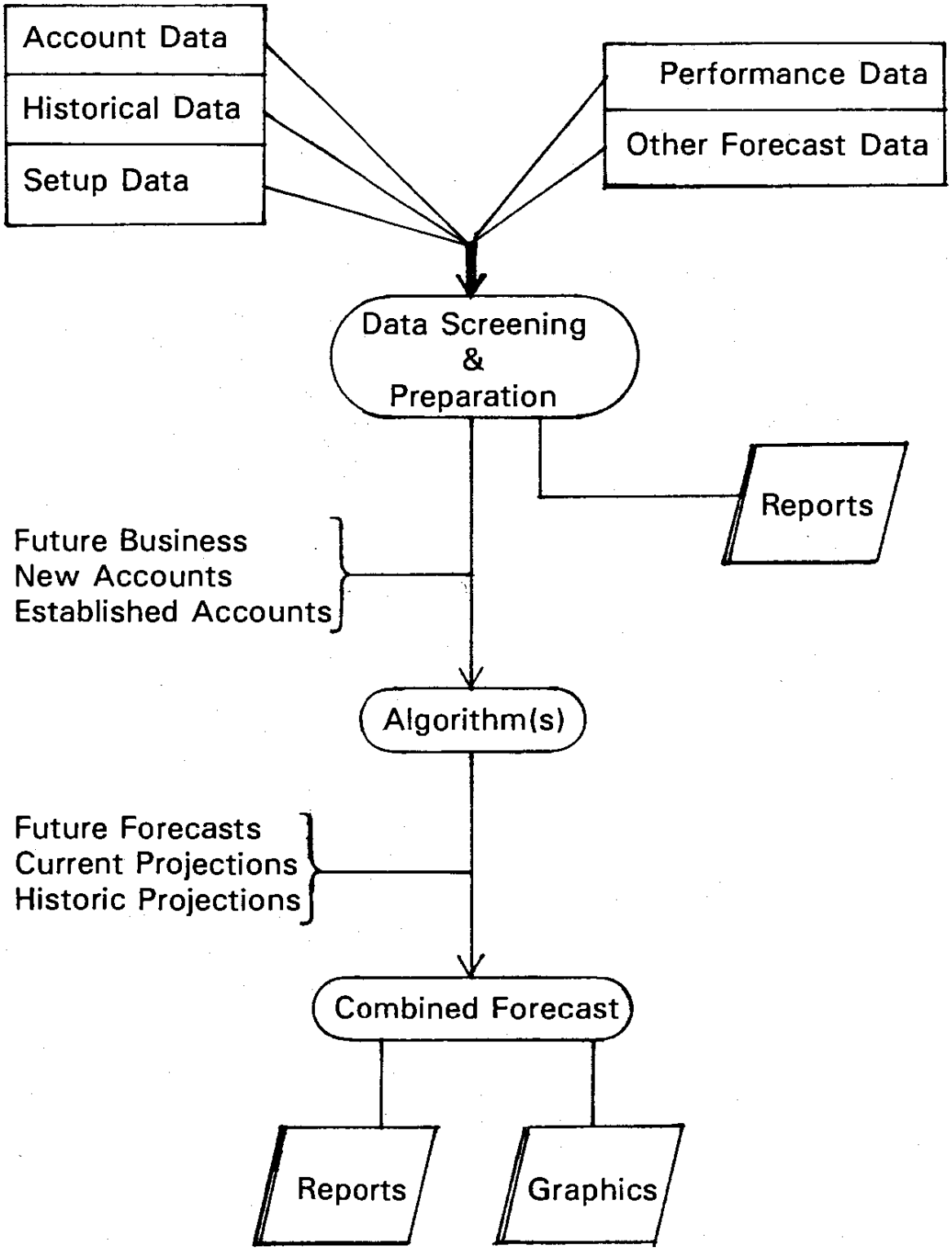
The correctness of the code and the system in which it is implemented was established in much the same way as the previously described algorithm. The table used to determine service days was a 'given'.

The appropriateness of the assumptions about the process being modeled may be questionable. Again the main justification for the use of the algorithm rests in face validity. We know the value of a contract is usually realized. We know that there are many contracts 'working' at any one time. In aggregate performance, the proportional allocation of the value of existing contracts should match the overall average income flow. In retrospective studies this algorithm tends to be very conservative. The value of contracts is pushed out in time.

HOW THE ALGORITHMS ARE USED:

The user interface consists of a series of menus developed under SAS/AF®. The default algorithm selection is made using the menu SETUP selection. The value selected in this screen is moved into a MACRO variable and is used to determine the 'sub-routine' to call.

Display 1. A Simplified Forecasting Application



Each main algorithm code module is brought into the appropriate location of various functional units of the system code as an 'external subroutine' using %INCLUDE statements. This structure was implemented to minimize the burden of maintaining algorithm code in multiple functional units.

The algorithm then processes each record (one per contract) and produces the desired numerical values for each month of the process year, and annual totals for each of the next 6 years. Excess 'value' beyond 6 years is accumulated in year six. The completed data set is then summarized and stored in a SAS data set. Subsequent processing formats and prints the report as specified for the selected menu.

The intermediary data forecast for each business segment is used subsequently to assemble comprehensive reports.

HOW THE ALGORITHMS WORK:

The tabular algorithm (Display 2) first puts the actual sales that have already occurred for a contract/record in the appropriate months. Next the forecast values are determined. In general the algorithm uses the current position in time relative to when the contract began, and the current remaining value. A spending rate is determined from a table look-up and applied to the value of the contract. For each subsequent month a spend rate is read from the table and applied to the contract value. This assures that the tabulated spend rate (or proportion of the contract value) is what is spent in each relative time period of the contract.

The value is 'spent' until there is no remaining value for the contract. Numerical results are retained for each month in the first year and on an annual basis for six subsequent years. Some protections are built into the code in case something goes wrong.

Note that the tables of spend rates are developed based on the TERM of the contract. Contracts that are not of the same TERM as the ones that are tabulated are 'promoted' to the next higher TERM that is in the table. Those that are of a greater TERM than 48 months are treated as if they were of TERM 48 months.

The proportional algorithm (Display 3) is the first algorithm that was developed. The general approach is to recognize the actual sales that have already occurred. Actuals (variables JAN FEB MAR etc.) in the process year are stored in the appropriate months (Forecast Month variables FM1, FM2, ... FM12) of the contract that is being processed in the current record. Next the forecast values are determined. In general the algorithm uses the current position in time relative to when the contract began, and the current remaining value. A spending rate is determined considering the remaining value and the remaining service days so that the same amount of money is spent on each service day (remaining value divided by service days left). The total value spent in each month then is directly proportional to the number of service days in that month which are obtained by table look-up.

As with the tabular algorithm, value is 'spent' until there is no remaining value for the contract. Numerical results are retained for each month in the first year and on an annual basis for six subsequent years. Some protections are built into the code in case something goes wrong.

Display 2. - Sales Forecast by Table

```
/* Allocation of sales based on a table*/
** have s_mo, m_po, begin, value, term,
   jan-dec actuals, curmon and year;
**put actual sales figures into forecast;
  if newsales eq 1 then process2=s_mo;
    else process2=curmon;
  do i=1 to process2;
    fsales(i)=asales(i);
    end;
  * calculate cumulative sales ;
  * calc remaindr and month position;
  remaindr=value-sum(of fm1-fm12, 0);
  ** zero added in to keep it from going missing;
  if remaindr le 0 then do;
     do j=process2+1 to 12;
       fsales(j)=0;
       end;
     return;
     end;
    else tempterm=term;
  ** forecasting loop ;
monthpr=m_po;
if tempterm ge 0 then do;
    if mod(tempterm,6) gt 0 then do ;
      * promote term to next level if not a 6er ;
      * pushes moneys into the future (conservative);
      tempterm=tempterm+(6-mod(tempterm,6));
      put 'TERM promoted for' ci= ;
      end;
  if tempterm eq 30 or tempterm eq 42
    then do;
      tempterm=tempterm+6;
      put 'TERM promoted by 6 for' ci=;
      end;
  if tempterm ge m_po then do;
    do i=process2+1 to 12 ;
     * get table value by term and month position ;
     set save.tablotab point=m_po;
     * write forecast value as tab-value times value ;
     select (tempterm);
       when (6) fsales(i)=t6*value;
       when (12) fsales(i)=t12*value;
       when (18) fsales(i)=t18*value;
       when (24) fsales(i)=t24*value;
       when (36) fsales(i)=t36*value;
       when (48) fsales(i)=t48*value;
       otherwise fsales(I)=t48*value;
       end;
   *put m_po= fsales(i)= remaindr=
         t6= t12= t18= t24= t36= t48=;
   * deduct from remaindr;
    if remaindr le fsales(i) then  do;
       fsales(i)=remaindr;
       remaindr=0;
       zeros=i+1; *zeroout rest of forecast months;
       do j=zeros to 12;
```

```
        fsales(j)=0;
        i=12; * get out of loop (tempterm do)
              prematurely  ;
        end;
      return;
      end;
   else  remaindr=remaindr-fsales(i);
  * update month position ;
    m_po=m_po+1;
  end;* loop;
end;       * tempterm ge  m_po;
else do i=process2+1 to 12; **this block moved
     from end;
     fsales(i)=0;
     end;
  end;       * if tempterm gt 0 ?;
  **get forecast for next 6 yrs (really 4 yrs
       due to table being 48 mos);
  if remaindr gt 0 and tempterm ge m_po then do;
    do i=1 to 6;
      monthpr=m_po;
      do j=1 to 12;
        set save.tablotab point=m_po; *get table
              value by term & mo ;
        select (tempterm);
          when (6) saveprct=t6;
          when (12) saveprct=t12;
          when (18) saveprct=t18;
          when (24) saveprct=t24;
          when (36) saveprct=t36;
          when (48) saveprct=t48;
          otherwise  saveprct=t48;
          end;

        if saveprct=1 then do;
          *end-of-table;
          saveamt=remaindr; *use up all of remaindr;
          nextyr(i)=nextyr(i)+saveamt;  *accumulate
            years worth;
          remaindr=0;
        *  j=12;          * premature end-of-loop;
        *  i=6;           * premature end-of-loop;
          return;
          end;
        else do;
          saveamt=saveprct*value;
          if remaindr lt saveamt then  do;
            * keep remaindr >=0;
            saveamt=remaindr;
            nextyr(i)=nextyr(i)+saveamt;
            remaindr=0;
            j=12;
            i=6;
            end;
          else do;
            nextyr(i)=nextyr(i)+saveamt;  *accumulate
              years worth;
            remaindr=remaindr-saveamt;
            m_po=m_po+1;
            end;
          end;
        end;     * incrementing m_po;
      *  put nxtyr1-nxtyr6= remaindr= m_po= t6= t12=
         t18= t24= t36= t48=;
      end;       *incrementing i ;
    monthpr=m_po;
    end;       * if tempterm ge  m_po
```

Display 3. - Sales as a Proportion of Contract Value

```
/* Proportional allocation of sales over sales days*/
  havsale=0; makfor=0;  daystogo=0;
  d1=0;      d0=0;      d4n=0;
  do i=1 to recs;
    offset=input("&year",4.)-1987 +i;
    * table starts at 1987 (1987-1986=1 - first obs);
    set save.saledays(drop=year ) point=offset;
    if _error_ then abort;
    if i=1 then do;
      d3=0;
      do j=1 to 12;
        fdays(j)=ldays(j);
      end;
    end;
    if 2 le i le (recs-1) then d3=d3+total;
      * 2nd to 2nd-to-last yrs;
    yrsub=i-1;
    if 0 lt yrsub lt 7
      then nextyr(yrsub) = total;
      else if yrsub gt 6
            then nextyr (6) = nextyr (6) + total;
  end;  * done getting sales days ;
  remaindr=value;

** Use actuals if available else calculate percentage;
 *   and calculate the days already used this year ;
    do i=1 to curmon ;
      fsales(i)=0;
      if newsales ne 1 and asales(i) gt 0 then do;
        fsales(i)=asales(i);
        remaindr=remaindr-asales(i);
        havsale+1; *flag indicates number of sales
          that have occurred ;
        d1=d1+fdays(i);
        end;
      else if havsale gt 0 then do;
        d1=d1+fdays(i);
        havsale+1;
        end;
      end;
   do i=curmon+1 to 12;
     if havsale ge 1 then do;
        * put different scenarios here...;
        ** is there  value to forecast? ...;
        if remaindr gt 0 then makfor=makfor+1;
        else do;
          do j=i to 12;
            fsales(j)=0;   *keep records that have
               been forecast or that have used up
               their values   ;
             end;
          return;
          end;* get next ci;
        * calculate days to go on contract;
        end;
      if makfor=1
      or (newsales eq 1 and i eq s_mo) then do;
        if (smo_term ge 12) then do;
          do j=12 to (i-havsale) by -1;
            d0=d0+fdays(j); * days first year from
            contract start;
            end;
          do j=12 to mod(smo_term,12)+1 by -1;
            d4n=d4n+ldays(j);
            * d4n - days not used final year;
            end;
          end;
        else if (smo_term le 12) then
          do j=(smo_term-1) to (i-havsale) by -1;
            d0=d0+fdays(j); * days first year from
            contract start;
```

```
        d4n=total; * days not used final year  ;
        end;
      daystogo=d0-d1+d3+total-d4n;
      if yrsub gt 0 then
        nextyr(yrsub)=nextyr(yrsub)-d4n;
      * total is the last value read
        from servicedays data;
      end;  *  makfor;
    * calculate forecasted services ;
    if daystogo gt 0 then do;
      fsales(i)=remaindr/daystogo * fdays(i);
      daystogo=daystogo-fdays(i);
      remaindr=remaindr-fsales(i);
      end;
    else  fsales(i)=0;
    end;
**   forecast routine for the next 6 years;
do i=1 to 6;
  if nextyr(i) gt 0 and daystogo gt 0 then do;
    savedays=nextyr(i);
    nextyr(i)=remaindr/daystogo * nextyr(i);
    daystogo=daystogo-savedays;
    remaindr=remaindr-nextyr(i);
    end;
  else do;
    if i=1 then put 'daystogo le 0? '_all_;
    nextyr(i)=0;
    end;
  end;
```

CONCLUSION:

A financial forecasting application which uses a table driven architecture has been described with particular reference to some of the basic forecasting algorithms, and their development and validation.

The proportional algorithm accounts for the differences in sales days in each month. It tends to push sales dollars into the future because of the flat spending rate. (Contracts generally are expected to have a tapering off of spending in the final months.) For new business it may be too aggressive because it will spend faster in the beginning than may be reasonable. This is probably offset by the failure to achieve the spending rates that would occur at the peak. This is a suitable approach for projecting aggregate contract behavior when specific information about individual contracts is not available.

The tabular algorithm is very simple and of limited value. It is most useful in a situation where information is known about large individual contracts. A specific spending performance can be tabulated as a function of the contract value. It will spend at those rates regardless of the previous history of spending for the contract. For a large aggregate of contracts it would work well if the tabulated spending performance represented the observed performance of the contracts in the aggregate. It does not account for differences in service performance in various months due to increased numbers of service days.

To summarize, the proportional algorithm is thought to be the best approach to use until more information can be obtained regarding contract spending performance. It should yield generally conservative performance. Once the ability to select specific algorithms for major contracts is implemented, the tabular algorithm can be used to good effect to force a specific spending pattern. For large contracts this could improve the reliability of the forecast.

REFERENCES:

1. Calvert, W.S. (1993), 'Forecasting Financial Performance for Long Term Care Facilities with SAS Software', *Proceedings of the Eighteenth Annual SAS Users Group International Conference*, 18, 313-318.

2. Fingerman, J. (1988) 'Forecasting as a Tool for Decision Making', *Proceedings of the Thirteenth Annual SAS Users Group International Conference*, 13, 187-192.

3. Fingerman, J. (1989) 'How to Establish a Forecasting System Using SAS Software', *Proceedings of the Fourteenth Annual SAS Users Group International Conference*, 14, 385-394.

4. Gilliland, M. & LaBella, J. (1993) 'Why forecasting is a Waste of Time: The Three Aphorisms', *Proceedings of the Eighteenth Annual SAS Users Group International Conference*, 18, 298-303.

The authors may be reached at:

Service Research, Inc.
401 Hull Place
Rockville, MD 20852

FAX 301.738.1041 or
VOICE 301.738.1024