「利用者履歴解析支援システム Preference Analyser の実装及び評価」

2S－6

犬束 敏信†　浜田 雅樹†
NTT ソフトウェア研究所‡

1 はじめに

本研究は，インタラクティブ・サービスに於ける利用者履歴（ログ）から利用者特性の解析を支援するシステムの構築を目指しており，筆者らは既にデータフロー図に基づいた解析記述モデル“Preference Analysis Model”を提案している [1].

本稿では，Preference Analysis Model による利用者履歴解析支援システム“Preference Analyser”（以下，PA）を，NTT が横須賀で行っている VOD 実験に適用した結果について述べる.

2 横須賀 VOD 実験での利用者履歴解析

2.1 横須賀 VOD 実験の概要

NTT では，インタラクティブ・サービスの技術確立と，マーケット・リサーチを目的とし，幾つかのマルチメディア実験を行なっており，横須賀地区における実験（以下，横須賀 VOD 実験）はその中のひとつである.

横須賀 VOD 実験では，CATV 利用者から選ばれた一般家庭および公共施設まで，光ファイバのアクセスネットワークを敷設している．この約 300 世帯（約 1,000 人の利用者）に対し，光ファイバを用いて，デジタル電話，CATV のデジタル伝送，および VOD サービスを提供している．実験は 1996 年 3 月から開始しており，システムの運用は NTT が担当している.

提供される VOD サービスは，MPEG2 による映画やスポーツの映像の配送，ショッピング，カラオケ，ネットワークを介して利用者同士がリアルタイムに対戦が可能なゲームなどから構成されている.

2.2 PA の目的と課題

インタラクティブ・サービスを提供するシステムでは，利用者の操作を逐一記録することが可能であるため，実際の行動データ（Revealed Preferece Data）を容易に取得することができる．横須賀 VOD 実験においても，実験参加者の同意により，利用者の操作をイベント・ログとして詳細に記録し，VOD サーバに蓄積している．この膨大なイベント・ログ中のデータを，カスタマ情報，及びコンテンツ情報と合わせて解析することによって利用者の個人の嗜好や，全体的な傾向などを調べ，インタラクティブ・サービス自身や他のサービスへフィードバック

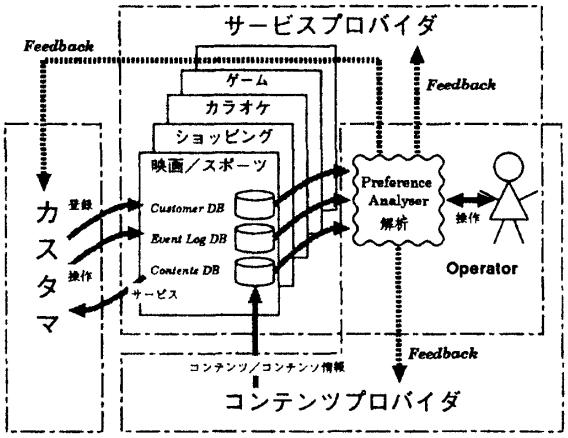

図 1: インタラクティブ・サービスと Preference Analyser

することにより，個人に合わせたサービスを提供することが期待されている．（図 1）

しかし，インタラクティブ・サービスは新しい形態のサービスであり，このような解析手法というものが未だ確立していない．横須賀 VOD 実験における PA の適用は，マーケット・データの取得だけでなく，解析手法そのものを模索することを目的としている.

以上の理由により，既存のツールが持っていない，

1. CATV 事業者や，コンテンツ・プロバイダ等の解析の非専門家が様々な解析方法を試行錯誤できること.
2. イベント・ログだけでなく総合的な解析，特にカスタマ情報やコンテンツ情報までを含めた解析を実現すること.

という要件を満たすツールの実現が必要となった．これらの要求に対し，次の特徴を持つシステム PA を開発した.

データフロー図（DFD）による図的記述 解析者は解析対象となるデータベース (操作履歴 DB，利用者情報 DB，コンテンツ情報 DB，等) のデータの解析手順を DFD エディタ上で視覚的に記述する．記述された解析手順はトランスレート機能により自動生成されるプログラムにより実行可能となる．図的記述のため，コンテンツ・プロバイダのような解析の非専門家にも容易に記述が可能となる.

状態イベント生成機能 ログ・データは利用者のシステムに対する操作（イベント）の記録であり，イベント発生時の利用者の状態の情報を含まない．システムの状態遷移図とこのイベントを照合することにより，状態を含む多角的な解析（どのような状況で利用者が他のサービスに遷移したか，等）が可能となる.

An implementation and evaluation of Preference Analyser
† Toshinobu INUZUKA, Masaki HAMADA
‡ NTT Software Laboratories

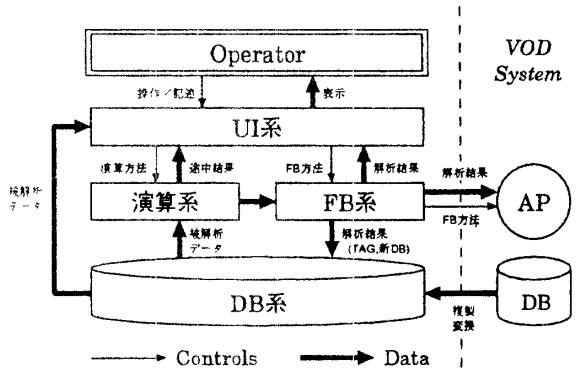

図 2: Preference Analyser の構成

3 システム構成

Preference Analyser を以下に示す 4 つのサブシステムにより構成した. (図 2)

DB 系 イベント・ログ, カスタマデータ, コンテンツデータを含むデータベースを管理する. イベント・ログは単発的なデータから状態を含んだデータへと変換を行なう.

UI 系 DFD を描くためのエディタである. 一般的なドロー・ツールと同様の操作でアイコンの配置／移動や, 記述した DFD のセーブ／ロードができる.

演算系 エディタにより記述された DFD を実行可能なプログラムに変換し, これを実行する. 今回の実装では, 実行プログラムとして Perl Script を用いている.

FB 系 解析データの出力を行なう. 出力先として, 結果データの表示, Microsoft Excel へのデータ出力, および VOD サービス・アプリケーションへのフィードバックがある.

本システムの動作環境については, 解析の記述を行なうデータフロー・エディタは Windows 95, または Windows NT3.51 でのみ動作するが, 生成された実行プログラムは Perl が使える任意の環境で実行させることができる.

Preference Analyser による解析の記述例と, 作成された Perl Script の例をそれぞれ図 3, 図 4に示す.

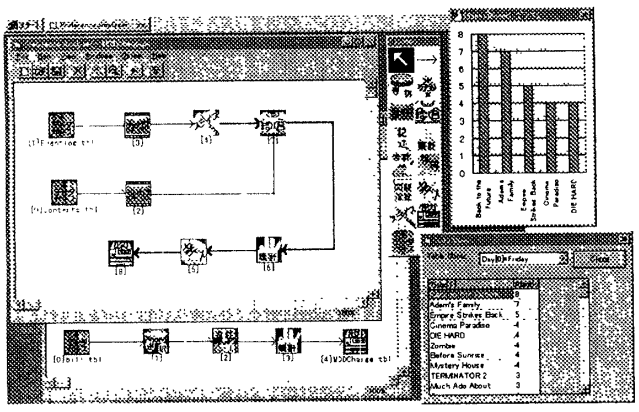

図 3: Preference Analyser による記述の例

4 評価

以下, 2.2節で述べた PA への要求条件の 1 つである非専門家による解析の記述の容易性について, 横須賀 VOD 実験に適用した結果について報告する.

1) 非専門家による実績 これまでに, PA を用いて数百種類の DFD が記述され, これらは毎月の解析に供されている. 担当者は, 統計解析のツールやプログラミングに関しては, 全くの素人であり, PA での記述に関しても, 操作説明のために 2 時間程度の準備学習を行なっただけである.

2) 記述能力 演算ノードとして, Joining, Arithmetic, Counting, Summation, Categorizing, Focusing, Filtering, Sorting, Slicing, Statistics, Correlation の 11 種類を予め用意した. これら, 11 種類の演算ノードのみで, 必要な解析を行なうことができた.

3) 記述量 ひとつの解析あたりの演算ノード数は, 約 9.7 個であった. ノードの入力アークの上限を 2 に設定したためであるが, 3 入力の記述を許容した場合, ひとつの解析あたりの演算ノード数は, 約 7.7 個となる. 一般的に人間が一目で把握できるオブジェクトの数は, 7 ± 2 個と言われている [2] ことからも, 視覚的了解性にも優れていることが分かる.

以上より, 非専門家が少ない演算ノードの組合せで, 必要な解析を容易に記述できることを確認した.

5 おわりに

本稿では, Preference Analysis Model に基づく, 利用履歴解析支援システム Preference Analyser を実装, 横須賀 VOD 実験へ適用し, プログラミングや市販解析ツールの知識を持たない素人にも容易に解析ができることを確認した. また, 離散的に記録されるイベントを状態を持ったイベントに変換することにより, 多様な解析を実現した. 今後は, WWW 等の VOD 以外のサービスでのログ解析へ適応していく予定である.

参考文献

[1] 犬束 敏信, 中西 弘毅, 荒野 高志. "Interactive System における Preference Analysis Model". In 情報処理学会研究報告 *95-DPS-70*, pages 45–50, 1995.

[2] George A. Miller. "The magical number seven, plus or minus two : some limits on our capacity for processing information". *Psychological Review*, 63(2):81–96, 1956.

```
for( $k=0 ; $k<$#slice1 ; $k++ ){
  seek( INFILE1 , $slice1[$k] , 0 );
  $sep1 = <INFILE1>;
  chop $sep1;
  reset e;     # to reset $entry
  reset d;     # to reset $data
  next if ( tell(INFILE) eq $slice
  while( <INFILE1> ){
    chop $_;
    @record1 = split(/ /,$_ , 100);
    $entry{$record1[0]}++;
    $data{$entry{$record1[0]}.$r
    last if ( tell(INFILE1) eq $sli
  }
  . . . . .
```

```
  . . . . .
while(<INFILE1>){
  if( /^@@@@@/ || /^=====/){
    push( @slice1 , $pointer1 );
  }
  $pointer1 = tell(INFILE1);
}
$pointer2 = tell(INFILE2);
while(<INFILE2>){
  if( /^@@@@@/ || /^=====/){
    push( @slice2 , $pointer2 );
  }
  $pointer2 = tell(INFILE2);
}
  . . . . .
```

図 4: 生成された実行プログラムの例