Deepthi Sehrawat al. International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 1, Issue 1, June 2014, pp. 78-81

Approach of Real Integrated Price language in Automation of Marketing Research Tool -Resolving Issue with Backup of CBR

Deepthi Sehrawat

Assistant Professor, CSE Dept. -ASET, Amity University Haryana, India

Abstract— **This research effort aims to automate the marketing research tool using real integrated price language and attempts to investigate its advantages over traditional expert systems approach. Real Integrated Price language work in various phases of tool Processing and Tool divided into the different parts on which the language show the capability:- a) Interface of Client Requirements Input b) URL Creator and Checker c) Streaming of the URL hit d) Crawling the web page e) Extract the information from the web Pages f) Creating the Final Report of the Extracted information g) Showing the All Data on the GUI interface. In this work input enter by the client is only manual process after this the complete process is process automatically by the tool and the language design for this is real integrated price language and it improve the efficiency of marketing research tool by 37 present and resolve the issue if phase by the system with case base reasoning approach.**

Keywords— **CBR, Marketing tool, Multi-agent system.**

I. INTRODUCTION

Real Integrated Price Language is divided into the different parts RIPL is a scripting language like assembly[1]. which is used to write the scripts for some specific needs. Generation of Dynamic URLs.

1. Crawling.
2. Extraction.
3. Report Generation.

But it is further subdivided in more parts. The collect the information like Hotel Name, Check-date, Checkout-date, Room Rate and Room Description with the help of this application i) Crawl the Web Pages from the different -2 sites with the help of URLs ii) Generation of URLS iii) Extract the information from the web Pages iii) Creating the Final Report of the Extracted information iv) Showing the All Data on the GUI interface[2]

II. COMMANDS IN REAL INTEGRATED PRICE LANGUAGE

Some Data type used in RIPL

a) Int :-
This instruction is used for declaring a integer variable.

b) Mov :-
This instruction is used to assign a value to a given variable.

c) Writestring :-
This instruction is used for copying string into HEAP.

d) readline :-
This instruction is used to read the specified Line into the RIPL interpreter file data buffer. This caching of data results in reduced disk access and faster processing of file data.

e) Writech :-
This instruction is used for accessing the HEAP at character level, to write value into it.

f) writenewline :-
This instruction is used to writing the specified Line into the a file interpreter file data buffer.

g) Readpage :-
This instruction is used to read the specified file data into the RIPL interpreter file data buffer.

h) If=0 :-
This instruction helps in performing conditional branching

i) if>0 :-
This instruction helps in performing conditional branching, if the first argument of this instruction evaluates to greater than zero.

j) If<0 :-
This instruction helps in performing conditional branching, if the first argument of this instruction evaluates to less than zero.

k) tostring :-
This instruction helps in performing the type conversion. To string can convert the value into string.

l) tono :-
This instruction helps in performing the type conversion. To int can convert the value into Integer.

m) print :-
This instruction is currently used for monitoring purpose

n) goto :-
This instruction is used for performing unconditional branching.

o) crawl :-
This instruction is used for crawl the webpages corresponding to a particular URL.

p) Readpage : -
This instruction is used to read the specified file data into the RIPL interpreter file
data buffer.

q) SetHead :-

This is instruction is similar in behavior to set the Cursor of currently used RIs.

r) movehead:-

This is instruction is similar in behavior to Move Cursor of currently used RIs.

s) scanfwd :-

This is instruction is to take the file pointer to the location in file where a given search
string exists.

t) gettext :-

This instruction starts copying from current file pointer position into HEAP at a
specified address until the terminating string string expression is found.

u) exit :-

This instruction is to terminate the program execution. As soon as this instruction is
encountered, the program execution terminates.

v) Broadcast :-

This instruction is used for printing the result which is on the Heap location.

III. PROCESSING AND EXECUTION OF STEPS IN REAL INTEGRATED PRICE LANGUAGE

Step 1

Interface of Client Requirements Input

Step 2

URL Creator and Checker

For collection of information like from web-link first of all url is generated according to SRS, url generated and maintain a file (Target page url may be from branded or commercial site according to specification)

```
20  int count
21  int found
30 int stringone
35 int reserve
36  int hotelid
40  writech  60000  13
41  writech  60002  10
50 int stringtwo
51 int startDay
52 int startMonth
53 int startYear
54 int ABP
55 writech  15000  48
56 int tempMonth
57 int tempMonthOne
100 varitostring "URL Out.txt" 8000
245 tono 0 hotelid
300 readline "HotelIds.txt" hotelid 7500 found
350 if=0  found  jump 550
355 tono 32 startDay
360 tono 64 startMonth
365 tono 96 startYear
366 mov  tempMonth  startMonth
370 tono 128 ABP
371 if=0 (count-ABP) jump 540
375 newdate startDay startMonth startYear  1
380 tostring startDay 2500
385 tostring startMonth 2509
390 tostring startYear  2518
391 tono 2509 tempMonthOne
425 readline "UrlTemplate.txt"  0  7000 found
430 varitonewline [8000]  [7000]
432 varitonewline [8000]  [96]
433 if<0  (9-tempMonth) jump 440
434 varitonewline [8000]  [15000]
440 varitonewline [8000]  [64]
442 varitonewline [8000]  [32]
445 readline "UrlTemplate.txt"  1  7000 found
450 varitonewline [8000]  [7000]
452 varitonewline [8000]  [2518]
453 if<0  (9-tempMonthOne) jump 455
454 varitonewline [8000]  [15000]
455 varitonewline [8000]  [2509]
456 varitonewline [8000]  [2500]
460 readline "UrlTemplate.txt"  2  7000 found
465 varitonewline [8000]  [7000]
470 varitonewline [8000]  [96]
471 if<0  (9-tempMonth) jump 473
472 varitonewline [8000]  [15000]
473 varitonewline [8000]  [64]
480 readline "UrlTemplate.txt"  3  7000 found
485 varitonewline [8000]  [7000]
490 varitonewline [8000]  [32]
492 readline "UrlTemplate.txt"  4  7000 found
494 varitonewline [8000]  [7000]
495 varitonewline [8000]  [2518]
496 if<0  (9-tempMonthOne) jump 498
497 varitonewline [8000]  [15000]
498 varitonewline [8000]  [2509]
499 readline "UrlTemplate.txt"  5  7000 found
500 varitonewline [8000]  [7000]
505 varitonewline  [8000]  [2500]
510 readline "UrlTemplate.txt"  6  7000 found
515 varitonewline [8000]  [7000]
518 varitonewline [8000]  [7300]
520 readline "UrlTemplate.txt"  7  7000 found
521 varitonewline [8000]  [7000]
525 varitonewline  [8000]  [60000]
526 varitonewline  [8000]  [60002]
527 copybytes 2500 32  8
528 copybytes 2509 64  8
529 copybytes 2518 96  8
530 mov  count count+1
535 goto 355
540 exit
550 varitonewline [8000] "Hotel-Not-Available"
560 exit
```

Fig-1. Dynamic URL generation code in RIPL

Step 3

Streaming of the URL hit:- Stream the url in particular order in which they hit

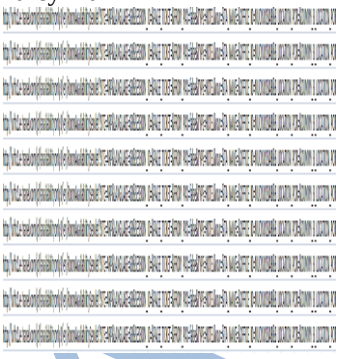

Fig-2. Streaming URL File

Step 4

Crawling the web page:- In this the target page is downloaded from the source in to particular location so that we can further use it for extraction the data

```
//  *********Reading the URL from a URL file
43   Writestring       [160]          10000
44   Writestring       "URLOut.txt"       12000
45   mov          iStringOneIndex    10000
46   mov          iStringTwoIndex    12000
47   mov          retaddress         50
49   goto         430
50   readline          "URLOut.txt"       urlno          33000          RdLineResult
51   if=0         RdLineResult       jump           79
//  **increment in the URLno for creating the Page no (1,2,3,4,5,6.....)
52   mov          PageNo             urlno+1
53   tostring          PageNo             1031
//  **Value of Pageno.*****************
54   mov          iStringOneIndex    1031
//  **File Extention **********************
55   mov          iStringTwoIndex    1025
//  **Assigning the value for returning by the function
56   mov          retaddress         58
// Jump to strcat() definition
57   goto         430
58   Writestring       [160]          7000
59   mov          iStringOneIndex    7000
60   mov          iStringTwoIndex    1031
61   mov          retaddress         64
62   print        [7000]
63   goto         430
//  50000 for null Value (Proxy)
64   crawl     7000     2000     33000     50000   PageSize   PageState
65   if=0         urlno          jump           83
66   mov          PageSize           PageSize/1024
67   print        PageSize
68   if=0         One            jump           75
69   mov          Attempt            Attempt+1
70   if=0         Attempt-4          jump           72
71   if<0         PageSize-SizeOne   jump           80
72   mov          Attempt            0
73   mov          urlno          urlno+1
74   goto         86
75   mov          One            One+1
76   mov          urlno          urlno+1
77   mov          SizeOne            PageSize
78   goto         86
79   Exit
80   print        "Crawl-Error"
81   print        [7000]
82   goto         64
85   goto         66
86   copybytes         7000          20000          32
87   goto    50
//  End of Crawling here
//  *****************function Definition for strcat() *****************
430  int          iSrcIndex
431  int          iSuffixIndex
432  int          iAscii
433  mov          iSrcIndex          iStringOneIndex
434  mov          iSuffixIndex       iStringTwoIndex
435  readch            iSrcIndex          iAscii
436  if=0         iAscii         jump           439
437  mov          iSrcIndex          iSrcIndex+1
438  goto         434
439  readch            iSuffixIndex       iAscii
440  writech           iSrcIndex          iAscii
441  if=0         iAscii         jump           445
442  mov          iSuffixIndex       iSuffixIndex+1
443  mov          iSrcIndex          iSrcIndex+1
444  goto         439
445  free         iSrcIndex
446  free         iSuffixIndex
447  free         iAscii
449  goto         retaddress
```

Fig-3. Crawling code in RIPL

Step 5

Extraction:- Extract the information from the web Pages

```
/ **********Variable Decleration
10  int     Found
11  int     PageNo
12  int     iStringOneIndex
13  int     iStringTwoIndex
14  int     urlno
15  int     PageSize
16  int     PageState
17  int     retaddress
18  int     Result
19  int     RdLineResult
20  int     SizeOne
21  int     One
22  int     Attempt
// **********End of variable Decleration
// ==========
// **********Witting the string for creating the price and for cookies
// ==========
38  mov     sizeOne     9999999
39  mov     PageNo      1
40  writestring   ".html"     1025
41  writestring   "jscript=1;path=/"  2000
42  writestring   size        15000
// *********Reading the URL from a URL file
// ==========
43  Writestring   [160]       10000
44  Writestring   "URLOut.txt"    12000
45  mov     iStringOneIndex   10000
46  mov     iStringTwoIndex   12000
47  mov     retaddress    50
49  goto    430
50  readline    "URLOut.txt"    urlno     33000     RdLineResult
51  if=0    RdLineResult    jump    79
// **increment in the URLno for creating the Page no (1,2,3,4,5,6.....)
52  mov     PageNo      urlno+1
53  tostring    PageNo      1031
// **Value of Pageno.**********
54  mov     iStringOneIndex   1031
// **File Extention **********
55  mov     iStringTwoIndex   1025
// **Assigning the value for returning by the function
56  mov     retaddress    58
// Jump to strcat() definition
57  goto    430
58  Writestring   [160]       7000
59  mov     iStringOneIndex   7000
60  mov     iStringTwoIndex   1031
61  mov     retaddress    64
62  print   [7000]
63  goto    430
// 50000 for null Value (Proxy)
64  crawl   7000  2000  33000  50000  PageSize  PageState
65  if=0    urlno     jump    83
66  mov     PageSize    PageSize/1024
67  print   PageSize
68  if=0    One       jump    75
69  mov     Attempt     Attempt+1
70  if=0    Attempt-4   jump    72
71  if<0    PageSize-SizeOne  jump    80
72  mov     Attempt     0
73  mov     urlno     urlno+1
74  goto    86
75  mov     One       One+1
76  mov     urlno     urlno+1
77  mov     SizeOne     PageSize
78  goto    86
79  Exit
80  print   "Crawl-Error"
81  print   [7000]
82  goto    64
85  goto    66
86  copybytes   7000      20000     32
87  goto  50
// End of Crawling here
```

Fig-4. Extraction code in RIPL

Step 6

Creating the Final Report of the extracted information.

Step 7

Showing the All Data on the GUI interface.

IV. ISSUE FACE BY RIPL

I. Non availability or logical error in URL calculation and collections[3].

II. Some time in RIPL the is logical error occurred and it provide un-relevant data and sometime URL is not calculated or not available of any new demands or previous one

III. Blocking of URL :- some time due to the abundant amount of hitting of a url for particular data ,the url's blocked for the accessing IP or Mac id, in this case we also loss the information

IV. Sometime page which is crawl down , we did not get any relevant data on the particular crawled page (may be site under construction , or some error problem of particular site etc)

V. CBR BACK-UP FOR RESOLVING ISSUE

CBR allows past solutions to be compiled in a reusable manner [4]. If a previoussolution's conditions of applicability can be abstracted and indexed then the CBRsystem can re-use these solutions. This ultimately saves computational time andallows the problem solver to avoid past mistakes. If the conditions that caused aprevious failure can be abstracted and indexed then the user can be saved the futility of repeating mistakes[5]. Human knowledge acquisition often involves the use of experiences and cases [6]. CBR therefore, offers a manner to model the way human's reason,making it easier to extract and store the expert's knowledge [7]. In RIPL, CBR notes the errors which are continuous in motion or regular or frequently occur.

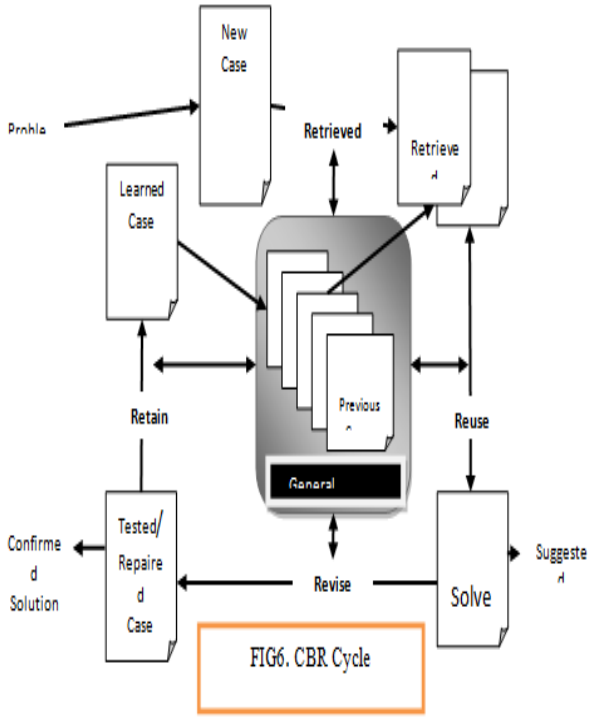

FIG6. CBR Cycle

VI. CONCLUSIONS

In this work input enter by the client is only manual process after this the complete process is process automatically by the tool and the language design for this is real integrated price language and it improve the efficiency of marketing research tool by 37 present and resolve the issue if phase by the system with case base reasoning approach

REFERENCES

[1]. Manoj, Rinkal, Vivek Jaglan, Mohit (2013) "Approach of Case Base Reasoning in Handling the Unavailable information Based on Real Integrated Price Language in a Marketing Research Tool" International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 3, May 2013

[2]. Vivek Jaglan (2011) "Approach of Agent Oriented Technology in Designing of Marketing Research Tools" International Journal of Computer Applications (ISSN 0975 – 8887)Volume 24– No.8, June 2011

[3]. [3] Vivek Jaglan, Vikas Dhankhar, S.Srinivasan, Manoj Kumar (2012) "A multi-agent based system for reduction of bullwhip effect in supply chain management" in Asian Journal Of Computer Science And Information Technology 2: 4 (2012) 82 – 88.

[4]. Aamodt, A. (1994a). Explanation-driven case-based reasoning.In Wess, S., Althoff, K., and Richter, M., editors, Topics in Case-Based Reasoning, pages 274– 288. Springer Verlag. .

[5]. Aamodt, A. (1994b). A knowledge representation system for integration of general and case-specific knowledge. In Proceedings from IEEE TAI-94, International Conference on Tools with Artificial Intelligence, pages 836–839, New Orleans, USA. IEEE Confence Proceeding.

[6]. Cohen, P. R. (1989). Evaluation and case-based reasoning. In Proceedings of a Workshop on Case-Based Reasoning, pages 168–172, Pensacola Beach, Florida, USA.

[7]. Endsley, M. R. (2000). Theoretical underpinnings of situation awareness: A critical review. In Endsley, M. R.

[8]. and Garland, D. J., editors, Situation Awareness Analysis and Measurement: Analysis and Measurement, pages 3–32

[9]. Grimnes, M. and Aamodt, A. (1996). A two layer case-based reasoning architecturefor medical image understanding. In Smith, I. and Faltings, B., editors, Advancesin case-based reasoning, Proceedings of EWCBR 1996, volume 1168 of Lecture Notes in Computer Science, pages 164–178. Springer Verlag.

