01 May 1994

Automation of Process Data Management

Stephen C. Chadwick

Automation of Process Data Management

Stephen C. Chadwick
Chemical Engineering Department

Abstract:

The project goal is to create an automated system for managing data that describes a chemical process. The data is both graphical, such as piping and instrumentation diagrams, and numerical, such as pipe schedule and diameter. AutoLisp and C++ programs were developed and tested to simplify the creation of the automated system.

Introduction:

The automated system is envisioned as follows: a computer user, most likely an engineer, will be at a PC. The engineer will decide the part of the process that needs to be modified. Perhaps the flow into a reactor should be increased or decreased or the pressure in a line should be changed or the piping in the plant should be rerouted. The engineer will make the graphical and numerical changes in AutoCad (version 12.0). AutoCad will communicate the numerical changes to Paradox, which is the attached data base, and the changes will update throughout the plant data. Essentially, this will allow plant configuration and parameters to be changed from a graphic user interface (GUI). This procedure is extremely beneficial because the piping and instrument diagrams (P&ID's) will always be current which is required by regulations set forth by the Occupational Safety and Health Administration (OSHA) and the Environmental Protection Agency (EPA). There will be fewer chances for transcription errors because both graphical and numerical data are updated using a single data management system. The automated data management system will eliminate transcription errors, save time, and make current drawings with correct plant parameters commonly available. Essentially, all changes will be registered in the computer and easily accessible to other personnel in the plant.

Background:

The C++ language is particularly valuable because it follows the object oriented paradigm and its flexibility. Object orientation is important to this project because of the way it handles classes through the use of structure. Because of these characteristics it is a useful tool when modeling plants. The following is an example of how C++ could be used to model pumps in a plant.

There is a class, a class of pumps. All of these pumps are in some way similar to each other. For example, they are used to move materials from one location to another and are rated by power requirements. The pumps have many common traits; these traits are grouped together into a superclass. However, there are various types of pumps (for example, centrifugal and positive displacement) that have different characteristics. These types of pumps would be subclasses (or children) of the superclass (or parent). In the object oriented paradigm, common characteristics of the superclass are inherited by each of its subclasses. All instances of a subclasses belong to the superclass. It is because of the inheritance of plant characteristics that object orientation is used.

Since C++ can handle these systems so well, a method was designed to interface with Paradox tables. This is possible based on the Paradox engine. This engine acts as an interpreter between C++ and Paradox. Essentially, programs can be written to search the Paradox tables and extract the desired data from the data base tables and place the information into the appropriate class structure of C++. It would also be possible to retrieve the data from Paradox, manipulate it in C++ and then return it back to Paradox. The data could also be manipulated and sent to different plant controllers, such as a distributed control system (DCS). The automated data management system has these capabilities.

Auto Lisp is the programming language of AutoCad. As the name indicates, Auto Lisp is designed to handle lists. These lists could range from plant sites down to the types of pumps to be used. The interface between AutoCad and Paradox is handled through Auto Lisp commands.

Project:

An Auto Lisp program was developed that would read strings from a computer text file based on a ordinal number given from a user input. (A listing of the source code for this program can be found in the attachments). The program works as follows: the computer user is asked by the program to input a number. The number specifies a line number in the text file where a string composed of substrings delimited by spaces resides. A sample file can be found in the attachments. Once the ordinal number is entered, the string in the file is printed at the bottom of the screen.

Also, a C++ program was developed that checks an Auto Lisp program. Auto Lisp counts parentheses as a first check. A C++ program was developed that reads an Auto Lisp program, counts the number of parentheses missing, and indicates whether or not a right or left parenthesis should be added. The program simplifies Auto Lisp program development because there is no compilation time under AutoCad, and the error analysis that is given by the program is much more detailed and easily understood by the user. (The source code can be found in the attachments).

The C++ source code for the Auto Lisp retrieve program is also included. The C++ retrieve function works in the same manner as the Auto Lisp function, but it is independent of any program platform. With the growing use of C++, due to the power and flexibility of the language, it is important whenever possible, to write the corresponding C++ code. Notice how much more compact the C++ code is when compared to the Auto Lisp code.

Applications:

At the end of an AutoCad session, the variables either created and/or used during the session cannot be saved if they exceed five in number. However, these variables can be written to an ASCII text file. The Auto Lisp retrieve program allows Auto Lisp variables to be stored in an ASCII text file and to be retrieved while working in an AutoCad session. The variables in the text file can be stored and retrieved in a subsequent session, as well. A text file called *names.txt* was populated with first, middle, and last names. The names in the data file represent any type of variable data created while working in an AutoCad session. The purpose of the retrieve function is to read the middle name on a specified line based on user input. This was chosen because retrieving the middle name involves characters before and after the desired substring, and these characters have to be taken into account. Often data in a system is surrounded by characters that are not needed. As long as the format of the data is known, the program can take into account potentially unwanted characters. This type of retrieve function is essential when working with large amounts of data, such as the automated data management system.

Acknowledgments:

At this time I would like to thank Dr. Book for all of his help and insight. Without his guidance I don't think that what I have accomplished would have been possible, and the knowledge that I have gained from this experience would never have been found. I would also like to thank Mr. Khandekar who was a fountain of information and helped me work out many bugs in my source codes, and also answered countless questions at all hours of the day with great enthusiasm.

Auto Lisp Program: Retrieving The Middle Name

```
(defun retrieve ()
       (setq refname "names.txt")
       (setq fil (findfile refname))
       (if fil
              (setq refname fil)
              (princ
              (strcat "\nPlease load the file "refname"."))
       )
(setq num (getint "Enter a number:"))
(setq f (open "names.txt" "r"))
(setq cnt 0)
(princ num)
(princ "\n")
  (while (< cnt num)
         (progn
              (princ num )
              (princ "\n")
```

```
                (princ cnt)
                (princ "\n")
                (setq d (read-line f))
                (princ d )
                (princ "\n")
                (setq cnt (+ cnt 1))
          )
 )
(setq a (read-line f))
(setq b (read-char f))
(princ "\n")
  ( while (/= b 32)
          (princ b)
          (setq b (read-char f))
 )
(princ"\n")
(setq mid " ")
(setq middle " ")
(setq b (read-char f))
  (while (/= b 32)
     (princ b)
     (princ mid)
     (setq middle (strcat mid (chr b)))
     (setq mid middle)
     (setq b (read-char f))
  )
(princ "\n")
(princ middle)
)
```

C++ Program: Function For Parentheses Check

```
//count.cpp
#include <fstream.h>
#include <conio.h>
#include <process.h>
void main(int argc, char* argv[])
        {
                if(argc != 2)
                  {
                   cerr<<"\nFormat: count filename.ext";
                   exit(-1);
                  }
                ifstream infile;
                infile.open ( argv[1] );
                if (!infile)
                  {
                   cerr<<"\nCan't open  "<<argv[1];
                   cout<<"\nPlease re-enter the command and filename.ext"<<endl;
                   exit(-1);
                  }
                int cnt1=0,cnt2=0;
                char ch;
                while (infile)
                  {
                   infile.get(ch);
                   if( ch = = ' ( ' ) cnt1++;
                   if( ch = = ' ) ' ) cnt2++;
                  }
                clrscr();
                cout<<"\nThere are  "<<cnt1<<"  ( parenthesis"<<endl;
                cout<<"There are  "<<cnt2<<"  ) parenthesis"<<endl;
                if(cnt1-cnt2 = = 0) cout<<"good job";
                else cout<<"Please try matching your ()'s again."<<endl;
     }
```

C++ Program: Retrieving The Middle Name

```
//This is C++ retrieve function (same as Autolisp's function, but in C++)
#include <fstream.h>
#include <conio.h>
#include <stdio.h>
void main()
 {
        int a=0;
        char ch;
        int counter=0;
        const int max=80;
        char buffer[max];
  clrscr();
  printf("This will retrieve the middle name");
  printf("\nEnter the line number of the name ");
  cin>>a;
  int b=a-1;
  ifstream infile("names.txt");
  while (infile && !(counter==b) )
        {
        infile.getline(buffer,max);
        counter++;
        }
  while(ch != ' ')
        {
        infile.get(ch);
        }
  infile.get(ch);
  cout<<"\nThe middle name of choice "<<a<<" is ";
  cout<<ch;
  while(ch != ' ')
        {
        infile.get(ch);
        cout<<ch;
        }
  cout<<endl;
 }
```

Names file:

```
David Michael Chadwick
Michael Ross Chadwick
Stephen Christian Chadwick
Mara Pageen Chadwick
Mary Francis Chadwick
Fred Stephen Chadwick
```