Ministry of Agriculture & Land Reclamation
Agricultural Research Center
Central Lab for Agricultural Expert Systems

The Design of the KSR Function Editor

TR/CLAES/238/2002.4

By

Eng.Mohammed El Helly
Eng.Mohammed Said
Eng.Ahmed Foad

April, 2002

1-Introduction

This Document show to the developer the design for the function editor and changes required in KSR to add this functionality.

2- Basic elements

The basic element of the function editor consists of

1- Function classes design.
2- Update in the Document class.

2.1 CFunction class

The function class consists of the following data members

a- **FunID:** function id.
b- Output: function result.
c- Body: expression, which will be evaluated, and its result stored in function output.

The following figure show Design for Cfunction class

```
class CFunction : public CObject
{
public:

        DECLARE_SERIAL(CFunction)
        CFunction();
        CFunction(CString ID);

        void SetOutput(CString Outputstr);
        void SetBody(CString Bodystr);
        CString GetBody() const;
        CString GetOutput() const;
        CString GetID();


        virtual void Serialize(CArchive& ar);

private:
         CString FunID;
        CString Output;
        CString Body;
};
```

Design of Cfunction class

2.2 Updates in the Document class

- **Frist step:**To deal with the function in the document a new member variable is declared

```
CFunClusterList m_FunctionList;
```

CfunClusterList: is a map contains two fields <FunGroup, FunList>
The map is declared as follows

```
typedef CTypedPtrMap<CMapStringToOb,CString,CObArray*>CFunClusterList;
```

- **Second step:** update the serialize method for document

```
void CKBEditorsDoc::Serialize(CArchive& ar)
{
        Version = 3;//after function
        if (ar.IsStoring())
        {
        ……….
        m_FunctionList.Serialize(ar);// store m_FunctionList.

        }
        else
        {
                switch(Version){
                        case 0:
                        case 1:  …………
                                …………
                                break;

                        case 2: ……………… break;
                        case 3:
                                ………………..
                                m_FunctionList.Serialize(ar);// loading m_FunctionList
                                break;

                        default:
                                AfxMessageBox ("Error: Invalid Document Format");
                                return;
                }

        }
}
```

3-Gammer of the function

The following figure shows BNF Grammar of the function

```
Statement → Input = Output
Input → cpt.prop
Output → Constant | Expr | Fun | cpt.prop
Expr → Expr + Term | Expr – Term | Term
Term → Term * Factor | Term / Factor | Factor
Factor → ( Expr ) | Digit | Fun | cpt.prop
Fun → Now | sin | sqrt | ………………………..
Digit → Nums [ Fraction ]
Nums → Num Num*
Num → 0 | 1 | 2 | ………….. | 9
Fraction → .digit
Constant → digit
```

BNF Grammar of the function

4- Additional Data Structure for Function Manipulation

4.1- FunToken Class

The FuncToken manipulate the function stream (tokenizing), and update the look ahead in the stream.

The class declaration show as the following:

```
class FunToken
{
public:
	FunToken();
	virtual ~FunToken();

	int GetStartPos();
	int GetLineNumber();
	void Reset(); //reset the static variable
	BOOL GetToken(CString &FunStream,CString &Token);
	BOOL GetConcept(CString &FunStream,CString &Token);
	BOOL GetProp(CString &FunStream,CString &Token);
	BOOL GetDate(CString &FunStream,CString &Token);

	void WriteToken(CString &FunStream);

private:
	static int StartPos; //the position in the function stream
	static int LineNumber; //the line number of the function
};
```

4-2 FunParser Class

This class used in parse the function and check on the computability of type between Output and Body of the function, then give error message if any error occur in the stream of the function.

The class declaration show as the following:

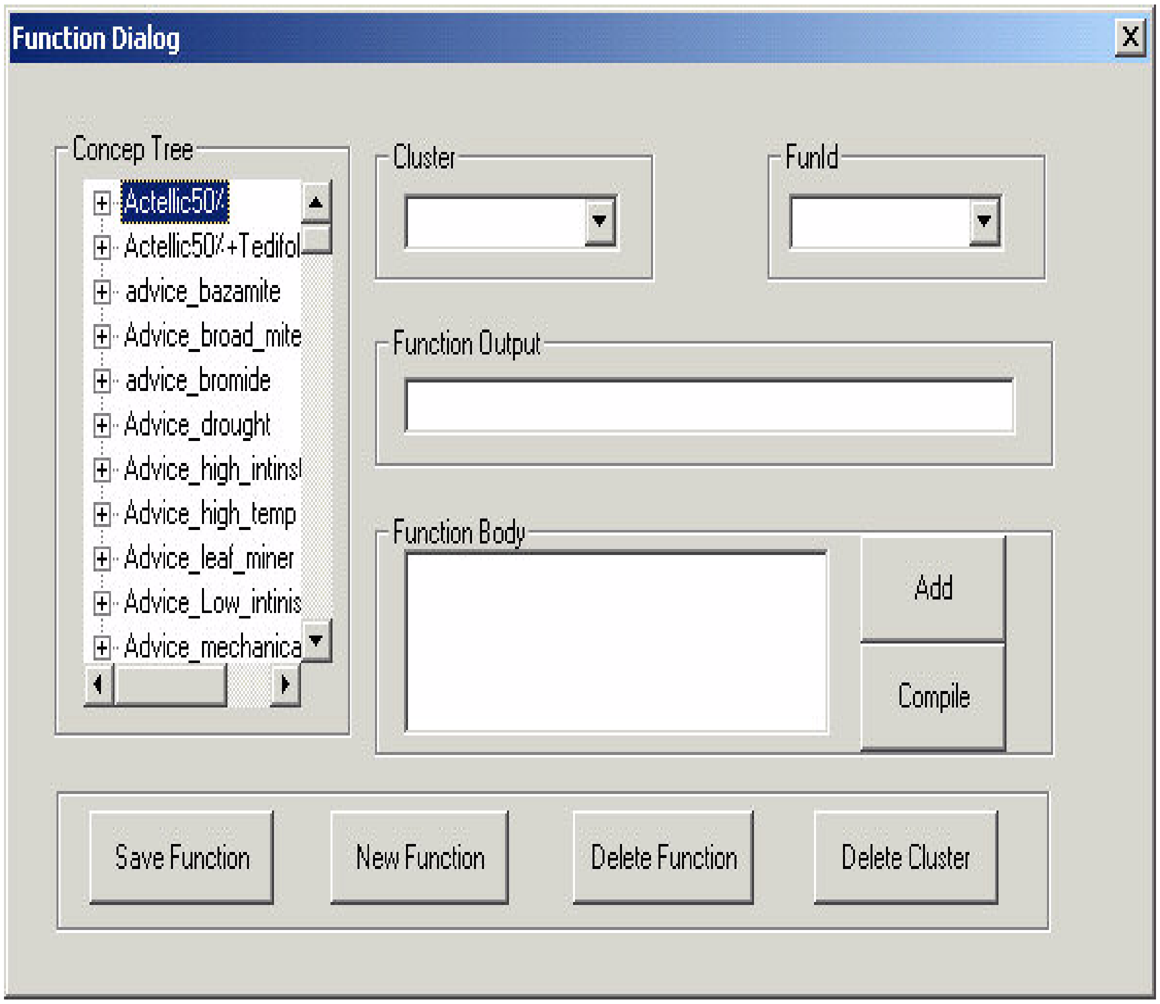
```
class FunParser
{
public:
        FunParser();
        virtual ~FunParser();

        BOOL ParseHeader(CString &FunStream,CString &Error);
        BOOL ParseBody(CString &FunStream,CString &Error);

private:
        FunToken TokenObj;//object from tokenizer
        int OutputType; //-1->initial 1->number 2->date
        int BodyType; //-1->initial 1->number 2->date
        int MathBuiltFun; //flag show there is math function(abs,...) or not

        void Reset();
        BOOL Parse_CPT_PROP(CString &FunStream,CString &Error);
        BOOL ParseBuilt(CString &FunStream,CString &Fun,CString &Error);
};
```

Beside that there is a set of helper functions declared as follow:

```
BOOL IsWhiteSpace(char ch); // return true if ch is a white space else false
BOOL IsAlpha(char ch);// return true if ch is an alphabetical character else false
BOOL IsNumber(char ch);// return true if ch is a numeric character else false
BOOL IsAt(char ch); //return true if ch is a '@' character else false
BOOL IsDot(char ch); //return true if ch is a '.' character else false
BOOL IsOperator(char ch);// return true if ch is an operator(+,-,…etc) character
// else false
BOOL IsSymbol(char ch);// return true if ch is a '(' OR ')' character else false
BOOL IsHash(char ch); //return true if ch is '#' character else false
```

5- Visual interface component design

The visual interface consists of combo boxes, textboxes, buttons, and events. The following Figure Demonstrate the proposed interface design:



5-1 Event associated with Function Dialogue

The above dialogue contains a lot of events; we will describe each of them in the following subsection.

5-1 On Load Dialogue

```
BOOL CFunDlg::OnInitDialog()
{
       // TODO: Add extra initialization here
         CDialog::OnInitDialog();
          SeeDoc(m_Doc);//to access document object
          DrawTree ();//build concept tree
          CenterWindow(GetDesktopWindow());
          return TRUE;
// return TRUE unless you set the focus to a control
  // EXCEPTION: OCX Property Pages should return FALSE}
}
```

5-2 On Click Save Function Button

```
void CFunDlg::OnSaveFun()
{
          CFunction  *Function1 = new CFunction;
          CString str,str1,error;
          FunParser obj;
          GetDlgItemText(IDC_Output1,str);
          GetDlgItemText(IDC_Body1,str);
          //compile function body and output
          if(error) then alert(error)
          else
          {
          if(an exist function has same function id) alert("message")
          else
              {
              set function1 output , body and id
              add function1 to current cluster
              add function1 id to FunList combo box
              clear Output edit box
              clear Body output box
              refresh cluster combo content
               save document
               }
          }


}
```

5-3 On Click New Function Button

```
void CFunDlg::OnNewFun()
{
// TODO: Add your control notification handler code here
   CString str="";
         SetDlgItemText(IDC_FunIDCOMBO,str);
         SetDlgItemText(IDC_Output1,str);//clear output edit box
         SetDlgItemText(IDC_Body1,str);
         SetDlgItemText(IDC_FunClusterCOMBO,str);
}
```

5-4 On Click Delete Function Button

```
void CFunDlg::OnDeleteFun()
{
         int Sel = m_FunCluster.GetCurSel();
         if (Sel < 0 ) return;
         CString str;
         m_FunCluster.GetLBText(Sel,str);
         m_Doc->m_FunctionList.Lookup(str,FunCluster);
         CString str1;
         GetDlgItemText(IDC_FunIDCOMBO,str1);
         CFunction *Function1;
         for(int i=0;i<FunCluster->GetSize();i++)
         {
                  Function1=(CFunction*)FunCluster->GetAt(i);
                  if(Function1->FunID==str1)
                  {
                           FunCluster->RemoveAt(i);
                           break;
                  }
         }
         delete Function1;
         SetDlgItemText(IDC_Output1,"");
         SetDlgItemText(IDC_Body1,"");
         OnSelchangeFunClusterCOMBO();
}
```

5-5 On Click Delete Cluster Button

```
void CFunDlg::OnDeleteCluster()
{
         int Sel = m_FunCluster.GetCurSel();
           if (Sel < 0 ) return;
           CString str1;
           m_FunCluster.GetLBText(Sel,str1);
           m_Doc->m_FunctionList.Lookup(str1,FunCluster);
           for(int j=FunCluster->GetSize()-1;j>0;j--)
           {
                     FunCluster->RemoveAt(j);
           }
           m_Doc->m_FunctionList.RemoveKey(str1);
           m_FunCluster.DeleteString(Sel);
           m_FunCluster.SetWindowText("");
           m_FunList.ResetContent();
}
```

5-6 On Click Add Button

```
void CFunDlg::OnAddBoy()
{
          HTREEITEM TreeItem = m_CptTree.GetSelectedItem();
          CString str1 = m_CptTree.GetItemText(TreeItem);
          HTREEITEM TreeItem1 = m_CptTree.GetParentItem(TreeItem);
          if (TreeItem1==NULL)return;
          CString str2 = m_CptTree.GetItemText(TreeItem1);
          CString str;
          CString m_str;
          GetDlgItemText(IDC_Body1,m_str);
          HTREEITEM TreeItem2 = m_CptTree.GetParentItem(TreeItem1);
          if (TreeItem2==NULL)
          {
                    str = "@" + str2 +"."+str1;
                    str = m_str + str;
                    SetDlgItemText(IDC_Body1,str);
                    return;
          }
          CString str3 = m_CptTree.GetItemText(TreeItem2);
          str1 = "\"" + str1 + "\"";
          str = "@" + str3 +"."+str2 +"=="+str1;
          str = m_str + str;
          SetDlgItemText(IDC_Body1,str);      ;
}
```

5-7 On Click Compile Button

```
void CFunDlg::OnCompileBody()
{
        CString error,str;
        FunParser obj;
        GetDlgItemText(IDC_Output1,str);
        if(obj.ParseHeader(str,error,&m_Doc->m_CptList)==false){
                AfxMessageBox(error);
                return;
        }
        GetDlgItemText(IDC_Body1,str);
        /****** Compile the Body *********/
        if(obj.ParseBody(str,error,&m_Doc->m_CptList)==false){
                AfxMessageBox(error);
                return;
        }
}
```

5-8 On Click Concept Tree

```
void CFunDlg::OnDblclkCpttree(NMHDR* pNMHDR, LRESULT* pResult)
{
        HTREEITEM TreeItem = m_CptTree.GetSelectedItem();
        CString str1 = m_CptTree.GetItemText(TreeItem);
        HTREEITEM TreeItem1 = m_CptTree.GetParentItem(TreeItem);
        if (TreeItem1==NULL)return;
        CString str2 = m_CptTree.GetItemText(TreeItem1);
        CString str;
        HTREEITEM TreeItem2 = m_CptTree.GetParentItem(TreeItem1);
        if (TreeItem2==NULL)
        {
                str = "@" + str2 +"."+str1;
                str = str;
                SetDlgItemText(IDC_Output1,str);
                return;
        }
        CString str3 = m_CptTree.GetItemText(TreeItem2);
        str1 = "\"" + str1 + "\"";
        str = "@" + str3 +"."+str2 +"=="+str1;
        str =str;
        SetDlgItemText(IDC_Output1,str);
        *pResult = 0;
}
```