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

# The Design of the Dependency Graph Editor (DGE)

**KSR Tool**

TR/CLAES/246/2002,9

By
Eng. Mohammed Yehia Hasan
Eng. Gamal Al-Shourbagy

**Sept. 2002**

# Table of contents

# 1. Introduction

**D**ependency **G**raph "**DG"** is a graph representation that depicts the dependencies that determine the property value. It consists of two main objects, which are nodes and links. There are many advantages of using this representation such as Avoid recalculations, Documentation, maintainability, Reusability, and Explanation facility. Also, there is another advantage to this representation which is the ability to reasoning by attribute name rather than the inference name. This reasoning is applied according to condition (i.e. conditional reasoning).

Thus, we are about to design DG tool, this tool will be a part of the KSR tool which used to assist the knowledge engineer to develop his knowledge as a graph like tree. The remaining sections would describe the detailed design for an editor that facilitates the constructions of this graph.

### *2. Overview Design for DG*

To meet the requirement specifications of DGE, which specified in [TR/CLAES/237/2002.4] report, figure 1 illustrates the overall structure of the DG as a part of the KSR. As shown in ***figure 1***, DG tool consists of DGEditor, DGReasoning, and DGQuery. All these components use the DGFile.

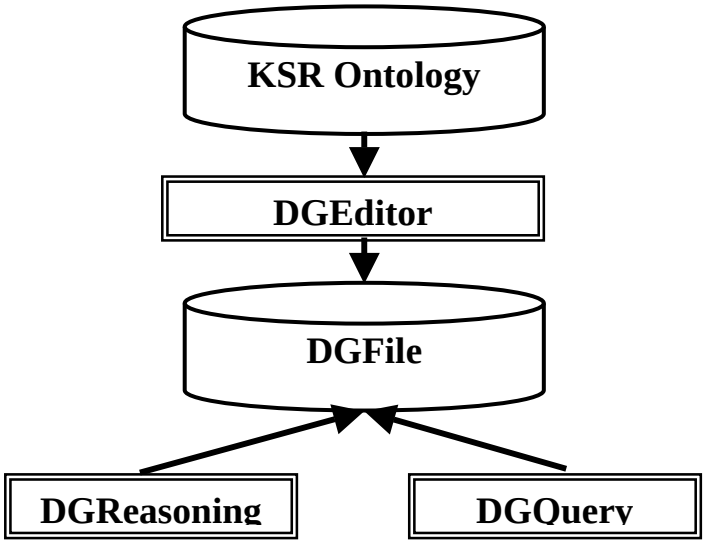



**Figure 1 Overall Structure of DG**

- KSR Ontology, is an external representation that contains the knowledge base.
- DGEditor, is a module that facilitates building DG.
- DGFile, is an output of DGEditor which is a part of KSR schema (i.e. the KSR schema should modified to be version 3).
- DGReasoning, is a forward chaining mechanism based on following up source of value of specified attribute.
- DGQuery, is a component that contains some modules that assist knowledge engineer to take an overview of the reasoning.

In the following sections we will illustrate those components in details.

### *3. KSR Ontology*

DG tool uses KSR ontology as a base structure of its knowledge base. So that, a little modifications in the KSR document ( Data Structures ) needed for the existence of the DG tool. This means that, the DG data structure is regards as a part of the KSR data structures.

The main class which represent the data structures of the KSR tool named "KBEditorsDoc" class. This class needs some modifications to support the DG data structures. ***Figure 2*** illustrates the detailed structures of the KSR Document after the modification. The statement "***CDGNodeList m_DGNodeList*** " would be added to the class. Also a little modifications to the operation serialize would be made. This modification would be explained in detailed in section 5.

```
class CKBEditorsDoc : public CDocument
{
protected: // create from serialization only
        CKBEditorsDoc();
        DECLARE_DYNCREATE(CKBEditorsDoc)

                //General Pointer
public:
        void InitDBWM();    CWMStructure* DB_WM;   DBDoc* DB_Doc;
        CString AppName, AppDir,  WorkingMemoryFile;
        CFunClusterList m_FunctionList;//ahmed fouaed
        CCluster *Cluster;    CConcept * Cpt;    CRule * Rule ;       CProperty * Prop;
        CImageListOb m_ImageListOb;    CDGNodeList m_DGNodeList;    CCptList m_TempCptList;
        CCptList m_CptList;   CCptList          m_CptListA;    CCptIDList  m_MapIDToCpt;
        long MaxConceptID;   long MaxRuleID;     BOOL Language;    CClusterList m_ClusterList;
        CInferenceList m_InferenceList;       CWMStructure* m_WM;          CTableList m_TableList;
        CXMLRPC * RPC;        CService * Service;        CServiceList m_ServiceList;
        CString DBName;      CString DBPath;    CKBDBList m_CKBDBList;
        CTblKeysList m_CTblKeysList;   CStringArray m_CBDKeysList;

public:
         //Temperary Variables
        CString CurrentCpt;   CString CurrentProp;
 //Operations
public:
         //Overrides            //ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CKBEditorsDoc)
public:
        virtual BOOL OnNewDocument();    virtual void Serialize(CArchive& ar);
        virtual void OnCloseDocument();       virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);
        //}}AFX_VIRTUAL
 //Implementation
public:
        virtual ~CKBEditorsDoc();
        void DecomposeString(CString str, CString &str1, CString &str2,const char * delemeter);
        void AbductAll(CString strCluster, CString strCpt,CString strProp);
        void AbductAll(CString strCluster, CString strCpt,CString strProp, CString strVal);
        void AddinTempList(CString str);    void SetDefaultInWM();
        BOOL IsSourceValue(CString ssSource, CProperty* Prop);
        BOOL IsSourceValue(CString ssSource, CString ssCpt, CString  ssProp);
        void InitWM();       void RemoveCptPropValFromWM(CConcept* Cpt,CString strVal);
        void PrintWM();     void PlayInference(CString strInf);
        void RemoveItemFromWM(CString Cpt, CString Prop);     CString GetVersion();
        BOOL SetDBAssociationToWM();       void AssertToWM(CString strCpt, CString strProp, CString
strVal);
         void SetDBValueToWM(CString strCpt, CString strProp, CString strVal);

#ifdef _DEBUG
        virtual void AssertValid() const;     virtual void Dump(CDumpContext& dc) const;
#endif
private:
        int Version;

 //Generated message map functions
protected:
        //{{AFX_MSG(CKBEditorsDoc)
         //NOTE - the ClassWizard will add and remove member functions here.
           //DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

***Figure 2*** Detailed structures of the KSR Document

## *4. DGFile Detailed Design*

As we mentioned in section 3, the overall data of the Dependency Graph would be a part of the KSR tool data file. The following subsections would be illustrates the two classes named "CDGNode and Clink". These classes represent the abstract data types for the node and link objects of the Dependency. Also the serialize operation for the objects of these classes would be present.

### *4.1. DG Node data structure*

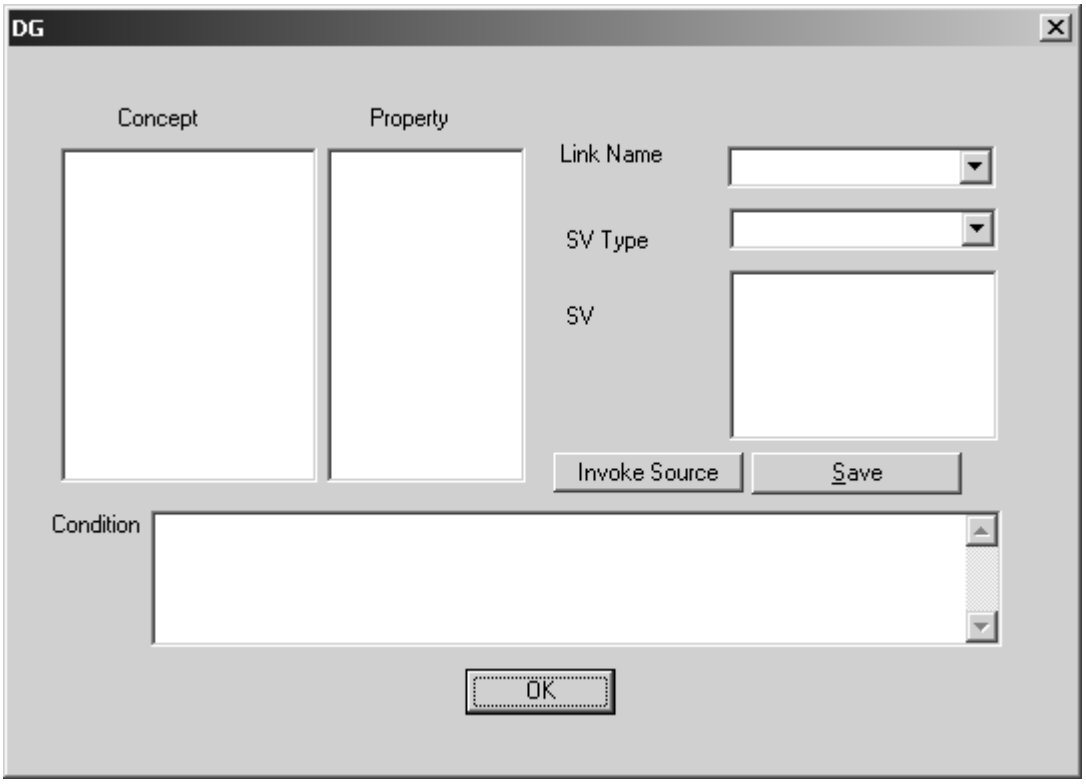
```
class CDGNode : public CObject
{
public:
        CDGNode();
        DECLARE_SERIAL(CDGNode)
        CString CptName;
        CString PropName;
        CLinkList m_LinkList;
public:
        virtual ~CDGNode();
        virtual void Serialize(CArchive& ar);
};
typedef CTypedPtrMap<CMapStringToOb,CString,CDGNode*>  CDGNodeList;
IMPLEMENT_SERIAL(CDGNode, CObject, VERSIONABLE_SCHEMA | 1 )

CDGNode::CDGNode()   {}

CDGNode::~CDGNode()  {}

void CDGNode::Serialize(CArchive& ar)
{
if (ar.IsStoring())
        {
        ar<< CptName;
        ar<< PropName;
        m_LinkList.Serialize (ar);
        }
        else
        {
                int nShema=ar.GetObjectSchema ();
        switch(nShema){
                        case 0:
                                ar>> CptName;
                                ar>> PropName;
                                m_LinkList.Serialize (ar);
                                break;

                        case 1:
                                ar>> CptName;
                                ar>> PropName;
                                m_LinkList.Serialize (ar);;
                                break;
                        default:
                                AfxMessageBox ("Error: Invalid Property Format");
                                return;
                }
        }
```

## *4.2 DG Link data structure*

```
class CLink : public CObject
{
public:
	CLink();
	DECLARE_SERIAL(CLink)
	CString LinkName;
	CString SVType;
	CString SV;
	CString Condition;
public:
	virtual ~CLink();
	virtual void Serialize(CArchive& ar);
};

typedef CTypedPtrMap<CMapStringToOb,CString,CLink*>  CLinkList;


IMPLEMENT_SERIAL(CLink, CObject, VERSIONABLE_SCHEMA | 1 )

CLink::CLink()
{
}

CLink::~CLink()
{
}

void CLink::Serialize(CArchive& ar)
{
if (ar.IsStoring())
	{
	ar<< LinkName;
	ar<< SVType;
	ar<< SV;
	ar<< Condition;

	}
	else
	{
		int nShema=ar.GetObjectSchema ();
	switch(nShema){
				case 0:
					ar>> LinkName;
					ar>> SVType;
					ar>> SV;
					ar>> Condition;
					break;

				case 1:
					ar>> LinkName;
					ar>> SVType;
					ar>> SV;
					ar>> Condition;
					break;
				default:
					AfxMessageBox ("Error: Invalid Property Format");
					return;
		}

	}
}
```

### *5. DGEditor Detailed Design*

The DGEditor is a module that facilitates building DG, it represents the user interface for the DG tool. This interface is a part of KSR tool user interface witch interact and use another user interface module within KSR tool. Figure 2 is the main form for the DG tool. As shown, the form would collect the data about each node from the knowledge engineer and allow him to interact with the node (attribute) source of value according to the node source type.




***Description of Dependency Graph Editor (DGE):***

As mention “DG” consists of two functions that are Nodes and Links. Each of these functions must be defined.

**Node Function:** is required to help users in defining their nodes. For each node, we need to define the following:

1. Node name, the attribute name from the ontology (concept. property)
2. Node source name (SV) and its type (SV Type), it will be selected from the list of sources already defined by the knowledge base.
3. Sub nodes, list of sub-nodes names or null. The sub nodes would be determined from the source of value of the node as its inputs.
4. Set of actions, the following action is proposed:
   a. Invocation of value source (value source editor), this action depends on the call message for the source dialog window.

**Link Function:** is the second part of the DGE. So, a link function is required to help users in defining their links. For each link, we need to define the following:

1. 'From Node'. A node that is currently defined is a "From Node".
2. 'To Node'. All the inputs of the node "From Node" are the "ToNodes", These are the nodes that the link point to. These nodes extracted from the "From Node" source of value.
3. Link Condition. This condition will determine whether the node "From Node" source of value would be visited or not.

***Global verification of the graph.***

A valid graph should satisfy the following conditions

1. A graph should start with a node whose source value is derived from Function, Table, or Rule (Cluster). This node represents the root of the graph.
2. A graph should be connected.
3. there is no direct or indirect cycling.
4. there is one and only one link between any two nodes.

Thus, in the following we will describe the DGEditor events:

## *5.1. OnInitDialog() Event*:

```
Begin
        CDialog::OnInitDialog();

   POSITION pos =m_Doc->m_CptList.GetStartPosition();
   CString   ConceptKey;
   CConcept *CptOj; // =new CConcept;
   while (pos != NULL)
   Begin
     m_Doc->m_CptList.GetNextAssoc( pos, ConceptKey, CptOj );
     m_ConceptLstCtrl.AddString(ConceptKey);
        End

        return TRUE;  // return TRUE unless you set the focus to a control
         //             EXCEPTION: OCX Property Pages should return FALSE
End

void CDGDlg::SeeDoc(CKBEditorsDoc*Doc)
Begin
        m_Doc = Doc;
End
```

## *5.2. OnSelchangelinknameCombo ()  event:*

```
Begin
        m_SVList.ResetContent();
        int Sel= m_SVTypeList.GetCurSel();
        if (Sel < 0 )
                return;
        CString SelectedSVType;
        m_SVTypeList.GetLBText(Sel,SelectedSVType);
        if (SelectedSVType == "Cluster")Begin
                POSITION posCluster =m_Doc->m_ClusterList.GetStartPosition();
                while (posCluster)
                Begin
                        CCluster * Cluster;
                        CString str ;
                        m_Doc->m_ClusterList.GetNextAssoc(posCluster,str,Cluster);
                        m_SVList.AddString(str);
                End
        End
        else if (SelectedSVType == "Function") Begin
                POSITION posCluster =m_Doc->m_FunctionList.GetStartPosition();
                while (posCluster)
                Begin
                        CObArray * FunCluster;
                        CString str ;
                        m_Doc->m_FunctionList.GetNextAssoc(posCluster,str,FunCluster);
                        m_SVList.AddString(str);
                End
        End
End
```

## *5.3. OnAddlink() Event:*

```
Begin
        int Sel= m_ConceptLstCtrl.GetCurSel ();
        if(Sel<0)Begin
                AfxMessageBox ("You Must Select Concept Name");
                return ;
        End
        CString SelectedConceptName;
        m_ConceptLstCtrl.GetText(Sel,SelectedConceptName);
         Sel= m_PropertyLstCtrl.GetCurSel ();
        if(Sel<0)Begin
                AfxMessageBox ("You Must Select Property Name");
                return ;
        End
        CString SelectedPropertyName;
        m_PropertyLstCtrl.GetText(Sel,SelectedPropertyName);
        CString NodeName;
        m_LinkList.GetWindowText (NodeName);
        if(NodeName""==)Begin
                AfxMessageBox ("You Must Enter Node Name");
        return;
        End
        Sel= m_SVTypeList.GetCurSel();
        if (Sel < 0 )Begin
        AfxMessageBox ("You Must Select Source Value Type");
                return;
        End
        CString SelectedSVType;
        m_SVTypeList.GetLBText(Sel,SelectedSVType);
        Sel= m_SVList.GetCurSel ();
        if(Sel<0)Begin
                AfxMessageBox ("You Must Select Source Value" );
                return ;
        End
        CString SelectedSourceValue;
        m_SVList.GetText(Sel,SelectedSourceValue);

        CString NodeCondition;
        m_ConditionEdit.GetWindowText (NodeCondition);
        if(NodeCondition""==)Begin
                AfxMessageBox ("You Must Enter Node Condition");
        return;
        End
        CLink *newClink=new CLink;
        newClink->Condition =NodeCondition;
        newClink->LinkName =NodeName;
        newClink->SV= SelectedSourceValue;
        newClink->SVType=SelectedSVType;
        CDGNode *DGNode;
        if(!m_Doc->m_DGNodeList.Lookup(SelectedConceptName+"-"+SelectedPropertyName,DGNode))Begin
                DGNode = new CDGNode;
                DGNode->CptName =SelectedConceptName;
                DGNode->PropName =SelectedPropertyName;
                m_Doc->m_DGNodeList.SetAt(SelectedConceptName+"-"+SelectedPropertyName,DGNode);
                DGNode->m_LinkList.SetAt(newClink->LinkName,newClink);
                m_LinkList.AddString(newClink->LinkName);
        EndelseBegin
                CLink *oldClink;
                if(DGNode->m_LinkList.Lookup(newClink->LinkName,oldClink))Begin
                        if(AfxMessageBox("Do you want to update existing Link ?",MB_YESNO) ==
IDYES)Begin
                                DGNode->m_LinkList.SetAt(newClink->LinkName,newClink);
                                AfxMessageBox("Operation done");
                        Endelse
                                AfxMessageBox("Operation canceld");
                EndelseBegin
                        DGNode->m_LinkList.SetAt(newClink->LinkName,newClink);
                        m_LinkList.AddString(newClink->LinkName);
                End
        End

End
```

## *5.4. OnSelchangePropertyLstCtrl() event:*

```
Begin
        m_LinkList.ResetContent();
        int Sel= m_ConceptLstCtrl.GetCurSel ();
        if(Sel<0)Begin
                AfxMessageBox ("You Must Select Concept Name");
                return ;
        End
        CString SelectedConceptName;
        m_ConceptLstCtrl.GetText(Sel,SelectedConceptName);
         Sel= m_PropertyLstCtrl.GetCurSel ();
        if(Sel<0)Begin
                AfxMessageBox ("You Must Select Property Name");
                return ;
        End
        CString SelectedPropertyName;
        m_PropertyLstCtrl.GetText(Sel,SelectedPropertyName);
        CDGNode *DGNode;
        if(m_Doc->m_DGNodeList.Lookup(SelectedConceptName+"-
"+SelectedPropertyName,DGNode))Begin
                POSITION pos = DGNode->m_LinkList.GetStartPosition();
                while(pos)Begin
                        CLink *newClink;
                        CString str;
                        DGNode->m_LinkList.GetNextAssoc(pos,str,newClink);
                        m_LinkList.AddString(str);
                End
        End
End
```

## *5.5. OnSelchangeConceptLstCtrl() event:*

```
Begin
        m_PropertyLstCtrl.ResetContent();
        int Sel= m_ConceptLstCtrl.GetCurSel();
        if (Sel < 0 )
                return;
        CString SelectedConcept;
        CConcept *CptOj;
        m_ConceptLstCtrl.GetText(Sel,SelectedConcept);
        m_Doc->m_CptList.Lookup(SelectedConcept,CptOj);

        CString PropertyKey;
        CProperty* PropOj;
        POSITION pos =CptOj->m_PropList.GetStartPosition();
        while (pos != NULL)
        Begin
                CptOj->m_PropList.GetNextAssoc( pos, PropertyKey, PropOj );
                m_PropertyLstCtrl.AddString(PropertyKey);
        End
End
```

## *5.6. OnSelchangeLinklist() event:*

```
Begin
        int Sel= m_ConceptLstCtrl.GetCurSel ();
        if(Sel<0)Begin
                return ;
        End
        CString SelectedConceptName;
        m_ConceptLstCtrl.GetText(Sel,SelectedConceptName);

        Sel= m_PropertyLstCtrl.GetCurSel ();
        if(Sel<0)Begin
                return ;
        End
        CString SelectedPropertyName;
        m_PropertyLstCtrl.GetText(Sel,SelectedPropertyName);
        CString LinkName;

       Sel= m_LinkList.GetCurSel ();
        if(Sel<0)Begin
                return ;
        End

        m_LinkList.GetLBText(Sel,LinkName);
        CDGNode *DGNode;
        if(!m_Doc->m_DGNodeList.Lookup(SelectedConceptName+"-
"+SelectedPropertyName,DGNode))
                return;

        CLink *oldClink;
        if(!DGNode->m_LinkList.Lookup(LinkName,oldClink))
                return;

        Sel= m_SVTypeList.FindString(-1,oldClink->SVType);
        m_SVTypeList.SetCurSel(Sel);
        CDGDlg::OnSelchangeCombo1();

        Sel= m_SVList.FindString(-1,oldClink->SV);
        m_SVList.SetCurSel(Sel);
        m_ConditionEdit.SetWindowText(oldClink->Condition);
End
```

## *6. DGReasoning Detailed Design*

Graph Reasoning is evaluated according to the method GetValue(). The algorithm of the GetValue(Concept.Property) is illustrates as the following:

## 6.1 GetValue() main function:

```
GetValue(C1.P1)
Begin
        If SV(C1.P1) = User then AskUser() exit()
        If SV(C1.P1) = DB   then invoke DB   exit()

        Get DGNode(C1.P1)
        For each link in DG.LinkList
        Begin
                If LinkCondition False Contiue()
                Else
                Begin
                        List = GetInputList(Link.SV)
                        If List is empty
                        Begin
                                Play Link.SV
                                Exit()
                        End
                        For each Node in List
                        Begin
                                GetValue(Node.(C.P))
                        End
                End
        End
Begin
```