THE APPLICATION OF GRAPH MODEL FOR AUTOMATION OF THE USER INTERFACE CONSTRUCTION

Elena Kudelko

Abstract: The ability of automatic graphic user interface construction is described. It is based on the building of user interface as reflection of the data domain logical definition. The submitted approach to development of the information system user interface enables dynamic adaptation of the system during their operation. This approach is used for creation of information systems based on CASE-system METAS.

Keywords: User interface, metadata, CASE-technology, dynamically adapted information systems, graph model.

ACM Classification Keywords: D.2.2 Software Engineering: Design Tools and Techniques – Computer-aided software engineering (CASE); G.2.2 Discrete Mathematics: Graph Theory – Graph algorithms.

Introduction

The aim of the working out of application user interface is the reflection of the inner structure of information system objects on the level of user understanding about data domain that means the determination of screen objects which let user co-operate with the information system (IS). User interface must include the set of screen forms with the help of which information input and editing can be possible, as well as the navigation system which let data domain objects be catalogued for speeding-up access to them. In this work the approach for automation of the logical description data domain reflection on user interface level, based on the use of graph model, is described. The working out of the models of user interface is made in the context of the creation of CASE-technology METAS [Lyadova, 2003] based on the metadata, which are multilevel and describe IS from different points [Ryzhkov, 2002]. User interface management, described in this work, is based on the metadata of presentation level, which are built on the base of logical level. Both levels can be represented as graphs.

The basic concepts of the logical level are entities, attributes, relation between entities and also instances of these concepts. *Entity* is a type of data domain objects which is characterized by the set of its *attributes* and *relations* with entities. For example, entity can be «Person» characterized by the qualities «Surname», «Name», «Birthday», which are attributes of this entity. A person must have an address that means that entity «Person» must be related with entity «Address». Metadata of the presentation level describe user interface elements: screen forms, controls of different type, navigation system of application. The idea of automatic creation of forms is based on the building of presentation level graph as a reflection of the logical level graph.

Screen Forms

Let us describe the graph of the logical model G_l, on the base of which we will build the graph of the presentation model G_{pr}. The nodes of the logical model graph are corresponded to entities of data domain; there are relations between entities, which are directed arcs in the graph of logical model:

$$G_l = (V_l, E_l), \text{ where } V_l = \{e_1, e_2, ..., e_n\},\ E_l = \{r_1, r_2, ..., r_m\},\ n,m \in N$$

$$r_i = (e_j, e_k), \text{ where } i = 1..m;\ j,k = 1..n$$

Incoming into the node *e* arcs mean relations of «1:M» type, in which entity *e* is on «M» side that means it is child entity. Outgoing arcs present relations of «1:M» type, in which entity *e* is on «1» side that means it is parent entity. Relations of «M:M» type are represented by two-forked arcs of graph G_l.

Each node of the presentation model graph G_{pr} is a form of some entity; arcs between nodes are possible transitions between forms (corresponding to arcs in the logical model graph); arcs are directional, direction is

given from the form reviewed to the forms which can be called from the current form:

$$G_{pr} = (V_{pr}, E_{pr}), \text{where } V_{pr} = \{f_1, f_2, ..., f_n\}, E_{pr} = \{r_1, r_2, ..., r_m\}, n, m \in N$$
$$r_i = (f_j, f_k), \text{where } i = 1..m; j, k = 1..n$$

Here pairs (e_j , e_k) and (f_j , f_k) are directed. That means graphs G_l and G_{pr} are oriented.

Graphs G_{pr} and G_l are, in general, multigraphs, in which cycles and loops are possible, and so, the incident nodes of the arc is not enough to identify this arc. That's why arcs must be marked with unique names, for $\forall e_1, e_2 \in V_l$ there must not be two arcs $(e_1, e_2) \in E_l$, having identical types and names. It is the same for graph G_{pr}.

The building process of the presentation model graph is in the reflection of the logical model graph G_l on the set of nodes and arcs of the presentation graph G_{pr}. In every moment the presentation model graph can be determined short. Let us see the building process of a new node *f* of graph G_{pr}.

The Elementary Graph of the Presentation Model

Node $f \in G_{pr}$ goes to some form for entity $e \in G_l$. This entity can be named the main entity form (we write $ME(f) = e$). So, we have: $(\forall f \in G_{pr})(\exists e \in G_l : ME(f) = e)$. The inverse proposition is also true. If we review the completely specified presentation model graph G_{pr}, i.e. the graph where there are't any undetermined nodes, then $(\forall e \in G_l)(\exists f \in G_{pr} : ME(f) = e)$. Arcs, incoming into node *e* of graph G_l, that means arcs for transition to parent entities in the relation of «1:M», «0..1:M», «0..1:1» type, are included into the presentation model graph (except two-forked graphs). Outgoing and two-forked arcs (arcs connecting reviewing entity with child entities) will be described later. Any arc in graph G_{pr} goes to some arc in graph G_l, i.e. $(\forall r_i = (f_j, f_k) \in E_{pr})(\exists r_i = (e_k, e_j) \in E_l : ME(f_j) = e_j, ME(f_k) = e_k)$.

Such graph model corresponds to the simplest case when for the reflection of every entity its own form is used and transitions between forms are realized by the relations of «1:M», «0..1:M», or «0..1:1» type, which exists between main entities of forms, in direction from child entity (from «M» side) to parent one (to «1» side). Let us view the example in Figure 1.

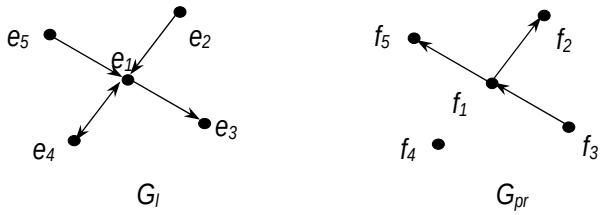

Figure 1. Reflection of the logical model graph on the presentation model graph

In graph G_l there are 5 nodes corresponding to 5 entities of data domain. Its own form is built for each of them: $e_i = ME(f_i)$, where $i = \overline{1,5}$. Entity e_1 has got two parent relations with entities e_2 and e_5, entity e_3 has got one parent – entity e_1. Therefore, there will be three arcs in graph G_{pr}, corresponding to the relations «1:M» of graph G_l. The building of graph G_{pr} goes according to gradual addition of nodes and its' arcs into sets V_{pr} and E_{pr} accordingly. That's why while building new node $f \in V_{pr}$ for node $e \in V_l$ it can happen that there is no node $f' \in V_{pr}$ for node $e' \in V_l$ yet, to which we must prolong arc from node *f*. Missing node f' must be built to avoid «hanging» arcs. We mark built node f' as indefinite («*udef*»). Such node has got one or several incoming arcs (in particular, arc $(f, f') \in E_{pr}$).

Attributes Reflection

Besides relations with other entities every entity is characterized by the set of its attributes. These attributes must be reflected by the elements of application user interface and must be included into the presentation model.

The creation of node *f* in graph G_{pr} is the reflection of logical model attributes to the controls of the presentation level. Any node $e \in V_l$ has got the set of attributes $Attr(e) = \{a_0, a_1, ..., a_n\}, n \in N$, where attribute a_0 is the key attribute of entity ($a_0 = key(e)$). In common case entity can have several key attributes (a compound key). But this situation can be taken to the viewed case by the addition of artificial key.The key attribute is used only on the logical level and is inaccessible for user on the presentation level.

Any non-key attribute a_i, where $i = \overline{1,n}$, can be either own attribute of the entity (the set of such attributes – $Attr_{own}(e)$), or an attribute, realizing the relation with parent entity, i.e. outer attribute ($Attr_{parent}(e)$). The key attribute is also in the set of own attributes. All own attributes have got a type $Type(a_i)$, which sets possible attribute values and operations, applicable to these values, and also the method of attribute values input and output. Every attribute a_i connecting with parent node $e' \in V_l$ is corresponding to any incoming in node *e* (but not two-forked) arc $(e', e) \in E_l$ ($rel(a_i) = (e', e)$). The reflection of such arcs in graph G_{pr} is set by the algorithms, described in the work. We can also speak about the type of such attribute. The type coincides with the type of the key attribute of parent entity e'.

Node *f* of graph G_{pr} includes in itself the set of the controls $AttrCtrl(f) = \{ac_1, ..., ac_n\}$, corresponding with attributes of entity $e = ME(f)$. The key attribute does not go into the presentation model. Each control will have the type, defining by the type of corresponding attribute or parent relation.

Compound Forms

Sometimes it is comfortable to include into the form information not only about one object of data domain but also the information connected with this object, i.e. the information about objects of several entities. Then each node of the presentation model graph will correspond to the subset of nodes of the logical model graph. The fulfillment of such reflection depends on the semantics of data domain and it can be done by user-administrator.

On the base of the built presentation model graph the extended graph can be built, nodes of which will be compound forms. Let us view the process of building such a graph. Let us have node *f* of graph G_{pr}, which corresponds to node *e* of graph G_l, i.e. $ME(f)=e$. Let's mark by $G_l(f)$ *some* set of nodes and arcs, complying with node *f* of the presentation model graph. In graph $G_l(f)$ every node is corresponded to node of the logical model graph, and arc is corresponded with the arc of the logical model graph. Now in the set of nodes there can be several nodes, corresponding to one and the same entity. The same is for arcs: in the set of arcs of graph $G_l(f)$ there can be several arcs, reflecting one and the same arc of the logical model graph. The set of nodes and arcs of graph $G_l(f)$ is marked correspondingly $V_l(f)$ and $E_l(f)$. Initially, the set of nodes of graph consists from one node, corresponding to entity *e*, and the arcs set is empty: $V_l(f) = \{ME(f)\}, E_l(f) = \phi$.

Let us view the arbitrary node *e* of the logical model graph G_l. Node *e* can be connected with the other nodes of graph G_l, i.e. there is a set of incident arcs to this node. Let us break this set on two subsets: the set of incoming arcs $E_{parent}(e) \subset E_l$ and the set of outgoing and two-forked arcs $E_{child}(e) \subset E_l$. The first set connects node *e* with the set of parent entities for this node

$$E_{parent}(e) = \{rp_1, ..., rp_n\}, n \in \{0\} \cup N, \text{where } rp_i = (e_i, e), e_i \in V_l, i = \overline{1,n},$$

and the second set – with the set of child entities

$$E_{child}(e) = \{rch_1, ..., rch_n\}, n \in \{0\} \cup N, \text{where } rch_i = (e, e_i), e_i \in V_l, i = \overline{1,n}.$$

In the example in Figure 1, relations (e_5, e_1) and (e_2, e_1) make up the set of parent relations for entity e_1, and relations (e_1, e_3) and (e_1, e_4) – the set of child relations.

Let us suppose that it is necessary to reflect on the form the information about the child entity of the main entity of form *f*. Let us examine the main entity *e* (*ME(f)=e)* and identical node to it e_{ch}, representing child entity for entity *e*. So, $\exists r_{ch} \in E_{child}(e) : r_{ch} = (e, e_{ch})$.

For including of node e_{ch} into graph $G_l(f)$ it is necessary to:

1. Include node e_{ch} into the set $V_l(f)$, and arc r_{ch} – into the set $E_l(f)$;
2. Include arc (f, f_{ch}), where $ME(f)=e$, $ME(f_{ch})=e_{ch}$, into the set E_{pr}. This arc will correspond to arc $(e, e_{ch}) \in E_l$.

To delete node e_{ch} from graph $G_l(f)$ it is necessary to fulfil the reverse transformation of the sets $V_l(f)$, $E_l(f)$ и E_{pr}:

1. Exclude arc (f, f_{ch}), where $ME(f) = e$, $ME(f_{ch})=e_{ch}$, from the set E_{pr};
2. Delete arc r_{ch} from the set $E_l(f)$, node e_{ch} must be excluded from the set $V_l(f)$.

Let us now have node e_p, incidental to the main entity *e* of form *f* (*ME(f)=e)* which is parent for it, i.e. $\exists r_p \in E_{parent}(e) : r_p = (e_p, e)$. Including and deleting node e_p from the set $V_l(f)$ occurs in the following way. For adding parent node to the main entity of the form it will be enough:

1. Include node e_p into the set $V_l(f)$, and arc r_p – into the set $E_l(f)$;
2. Include arcs, corresponding to the relations between node e_p and its parent nodes, i.e. for $\forall r \in E_{parent}(e_p)$ we include arc (f, f_r), where $ME(f_r) = e_r, r = (e_r, e_p)$;
3. Delete arc (f, f_p), where $ME(f)=e$, $ME(f_p)=e_p$, from the set E_{pr} of graph G_{rp}. This arc corresponds to arc $(e_p, e) \in E_l$.

Deleting of parent node from the main entity of the form includes the following steps:

1. Include arc (f, f_p), where $ME(f)=e$, $ME(f_p)=e_p$, into the set E_{pr} of graph G_{rp}. This arc corresponds to arc $(e_p, e) \in E_l$;
2. Exclude arcs corresponding with relations of node e_p with parent nodes, i.e. for $\forall r \in E_{parent}(e_p)$ we delete corresponding to it arc (f, f_r), where $ME(f_r) = e_r, r = (e_r, e_p)$;
3. Delete arc r_p from the set $E_l(f)$, node e_p must be excluded from the set $V_l(f)$.

Let us give the example, showing described operations of including of nodes into the form structure for entity *e*.

In Figure 2a the logical model graph is shown. Let us discuss the fragment of the presentation model graph, built for the form entity *e*. Figure 2b shows the simplest presentation model graph, built according the logical model graph. In Figure 2c there is the form structure *f* after including into it child relation (e, ech_1) and, so, adding relation (f, fch_1) into graph G_{pr}. In Figure 2d parent relation (ep_2, e) is included into the form structure, besides in graph G_{pr} connections (f, fp_{21}), (f, fp_{22}) are added and connection (f, fp_2) is deleted.

Similarly we can view adding and deleting operations for other entities, including in subgraph $G_l(f)$. In order to view these operations in details, we introduce the definition of parent entity of level *n*.

Let us have entity $e \in V_l$. The parent of the first level for this entity can be any entity, relation of which with entity *e* is in set $E_{parent}(e)$. If entity e_p^{n-1} is a parent of level *n-1* for entity *e*, then entity $e_p^n : (e_p^n, e_p^{n-1}) \in E_{parent}(e_p^{n-1})$ will be a parent of level *n* for entity *e*. The parent set of level *n* for entity *e* we will mark $E_{parent}^n(e)$. $E_{parent}^1(e) = E_{parent}(e)$.

For example, in Figure 2a $E_{parent}^3(ech_2) = \{(ep_{21}, ep_2), (ep_{22}, ep_2)\}$.

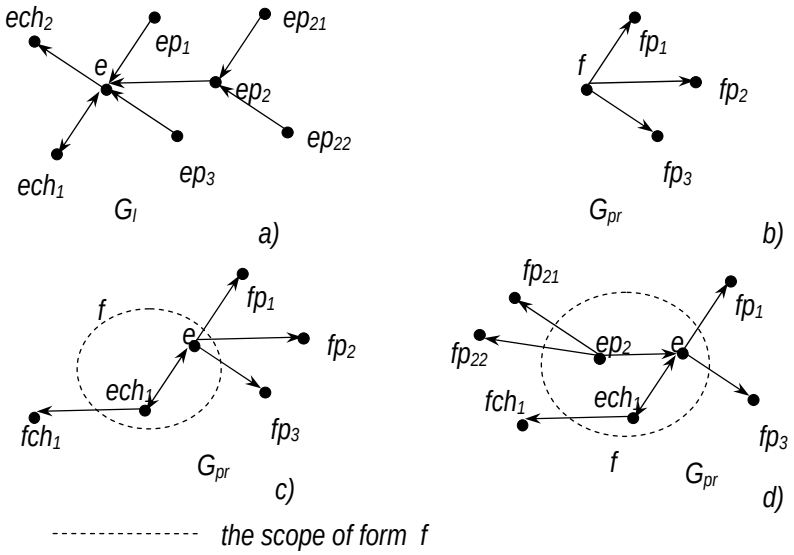

Figure 2. Including parent and child entities into the form structure

The discussed above operation of the addition of node $e_p \in E_{parent}(e)$ to the set $V_l(f)$ was described for e_p, which is the parent of the first level for node e. For node e_p, including in the set $V_l(f)$, and also for any node from the set $V_l(f)$, which is a parent of arbitrary level for node e, we can define the similar operations of deleting\adding parent and child entities.

We can examine node $f \in G_{pr}$ and entity $e=ME(f)$. We will take any node e', which is parent of arbitrary level of node e. We must notice that described below algorithms will be right for the case, when $e' = e$.

Let us have node e_{ch}, which is incidental to node e' of form f and which is child for her:

$$r_{ch} \in E_{child}(e') : r_{ch} = (e', e_{ch}).$$

For including node e_{ch} into graph $G_l(f)$ it is necessary to execute the following algorithm:

1. Add node e_{ch} into the set $V_l(f)$, and arc r_{ch} – into the set $E_l(f)$;
2. Add arc (f, f_{ch}), where $ME(f)=e$, $ME(f_{ch})=e_{ch}$, into the set E_{pr}. This arc corresponds to arc $(e', e_{ch}) \in E_l$;

In order to delete node e_{ch} from graph $G_l(f)$ it is necessary to fulfill the reverse transformation of the sets $V_l(f)$, $E_l(f)$ and E_{pr}:

1. Exclude arc (f, f_{ch}), where $ME(f)=e$, $ME(f_{ch})=e_{ch}$, from the set E_{pr};
2. Delete arc r_{ch} from the set $E_l(f)$, node e_{ch} must be deleted from the set $V_l(f)$.

Let us now have node e_p, incidentical to node e' of form f and which is parent for it, i.e. $r_p \in E_{parent}(e') : r_p = (e_p, e')$.

Addition of parent node into the form structure is in the following:

1. Add node e_p into the set $V_l(f)$, and arc r_p – into the set $E_l(f)$;
2. Add arcs, corresponding to relations between node e_p and parent nodes, i.e. for $\forall r \in E_{parent}(e_p)$ we add (f, f_r), where $ME(f_r) = e_r, r = (e_r, e_p)$;
3. Delete arc (f, f_p), where $ME(f)=e$, $ME(f_p)=e_p$, from the set E_{pr} of graph G_{rp}. This arc corresponds to arc $(e_p, e') \in E_l$.

Deleting of parent node from form structure consists of the following steps:

1. Add arc (f, f_p), where $ME(f)=e$, $ME(f_p)=e_p$, into the set E_{pr} of graph G_{pr}. This arc corresponds to arc $(e_p, e') \in E_l$;
2. Delete arcs, corresponding to the relations between node e_p and parent nodes, i.e. for $\forall r \in E_{parent}(e_p)$ we delete corresponding arc to it (f, f_r), where $ME(f_r) = e_r, r = (e_r, e_p)$;
3. Delete arc r_p from the set $E_l(f)$;
4. Fulfill cascading deleting of all parent nodes which are in $G_l(f)$, i.e. apply recursively the algorithm to every node $e_p^i \in V_l(f) : e_p^i \in E_{parent}(e_p)$;
5. Delete cascadely all child relations of node e_p, which are in $G_l(f)$, i.e. we must apply the deleting algorithm of child node to every node $e_{ch}^i \in V_l(f) : e_{ch}^i \in E_{child}(e_p)$;
6. Node e_p must be excluded from the set $V_l(f)$.

Described above the rules of adding nodes can be used in series until the expansion of graph $G_l(f)$ is possible, i.e. till set nodes $V_l(f)$ have parent and child nodes and relations that can be included into graph $G_l(f)$. Node e (and incidental to it arc) can be included into $G_l(f)$, if the graph does not have the path, including the same consecution of nodes and arcs, starting in the root, as well as the way from root node up to including node e.

So the repeated bringing in of one and the same path to graph $G_l(f)$ is impossible. And graph $G_l(f)$ is a tree in which the root is a node, corresponding to the main entity of form.

Attributes Reflection in Compound Forms

Including of parent node $e \in V_l$ into graph $G_l(f)$ of any node $f \in V_{pr}$ is a reflection of the attributes of the logical model on the controls of the presentation level. Deleting node e from graph $G_l(f)$ is a deleting of corresponding controls. Node f of graph G_{pr} includes a set of controls $AttrCtrl(f) = \{ac_1, ..., ac_m\}, m \in N$, corresponding to attributes of entities of the set $V_l(f)$. So,

$$AttrCtrl(f) = \bigcup_{e \in V_l(f)} AttrCtrl(f, e).$$

When adding new parent node e_p from relation (e_p, e'), where $e' \in V_l(f)$, to set $V_l(f)$:

1. The element corresponding to parent relation (e_p, e'), i.e. $ac \leftrightarrow a \in Attr(e') : rel(a) = (e_p, e')$ is deleted from the set of controls.
2. We add controls ac into the set of controls of node f, corresponding to all non-key attributes a of entity e_p, i.e. $\forall a \in Attr(e') : a \neq key(e_p)$ into the set $AttrCtrl(f)$ we put $ac \leftrightarrow a$.

When deleting parent node e_p, taking part in relation $(e_p, e') \in E_l(f)$ from the set $V_l(f)$:

1. We delete all controls, corresponding to entity e_p from the set of controls.
2. Into set $AttrCtrl(f)$ we add $ac \leftrightarrow a : rel(a) = (e_p, e')$.

Entity Tree

A tree is a building of hierarchy on the set of entities and relations between them. Together with the set of forms the set of tree nodes must provide access to any entity. The described structure is a tree only on a user's screen (its' name comes from here). From the point of view of the structure it is an oriented graph $G_T = (V_T, E_T)$, maybe

with cycles. The set of nodes $V_T = (nd_1, nd_2, ..., nd_n)$ includes two types of nodes (two subsets). The first subset $V_T^g \subset V_T$ consists of grouping nodes, the second subset $V_T^o \subset V_T$ has got object nodes. Any object node corresponds to the entity of the logical model and so, the form of the presentation level. Correspondence of object nodes and nodes of the logical model graph can be given as function $Ent: V_T^o \to V_l$. Then $(\forall nd \in V_T^o)(\exists e \in V_l : Ent(nd) = e \wedge \exists f \in V_{pr} : ME(f) = e)$. In that way, we can draw arc *(nd, f)* from object node *nd* to corresponding node-form *f*. The set of such arcs forms the additional set (let us call it E_{ext}), connecting two graphs G_{pr} and G_T. Nodes of the logical model do not correspond to nodes of the entity V_T^g . Such nodes are only created for convenient reflection of the information on the user's screen. There are arcs between nodes of the set V_T. Arcs can exist as between nodes of one subset (V_T^o или V_T^g), as also between nodes of different subsets. There is one group node in the tree from which the tree scanning starts. Such node can be marked as a *root*. The root node does not have incoming arcs. Any node of the set V_T is reachable from the root, i.e. between any node of the set V_T and root node there is a path.

The path between two object nodes can correspond to subgraph of the logical model which includes some sequence of arcs and nodes, which are between entities that are reflected by two viewed nodes.

For example, in Figure 3, graph of the logical model G_l can be corresponded to the graph of the object tree G_T. In the tree graph object nodes e_1, e_2, e_3 are named corresponding to the names of nodes-entities of graph G_l, which they correspond to. Arc $(e_1, e_3) \in G_T$ is corresponded to the path $\langle e_1, r_{14}, e_4, r_{43}, e_3 \rangle$, arc (e_2, e_2) – to the path $\langle e_2, r_{22}, e_2 \rangle$, arc (e_2, e_3) – to the path $\langle e_2, r_{23}, e_3 \rangle$.

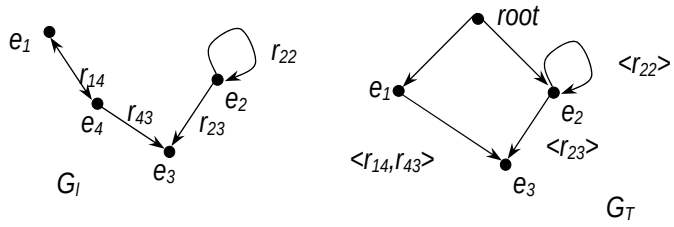

Figure 3. Example of entity tree

Let us view two object nodes $nd, nd' \in V_T^o : Ent(nd) = e, Ent(nd') = e'$, between which there is a path $\langle nd_1, (nd_1, nd_2), nd_2, (nd_2, nd_3), ..., (nd_{n-1}, nd_n), nd_n \rangle, nd_1 = nd, nd_n = nd', n \in N$. This path is corresponded to the path of the logical model graph $\langle e_1, (e_1, e_2), e_2, (e_2, e_3), ..., (e_{m-1}, e_m), e_m \rangle, e_1 = e, e_m = e', m \in N$.

Every node in graph G_T can be reachable from the root by different paths, and each of these paths can be concerned with the path in the logical model graph. While building the tree on the user's screen, i.e. during data loading, only important in this context relations are considered. While building a tree part of the ways can be reflected on the user's screen and the other part serve for assignment of additional dependence. Let us say that additional ways of the logical model connect with the way of graph G_T , consisting from one arc. For unification of the ways assignment we can also take that visual paths in graph G_T do not have corresponding paths in the logical model graph and all entities relations specify by non-visual additional connections.

Let us make it clear on the example. There is a fragment of graph G_T, shown in Figure 4. Here while building the branch which includes nodes *root, nd1, nd2, nd3*, for loading nodes of type nd_3 relations (nd_3, nd_2) and (nd_3, nd_1) are used, but while loading the branch which goes over nodes *root, nd4, nd5, nd2, nd3*, relations (nd_2, nd_5) and (nd_2, nd_4) are used, while loading nodes of type nd_2 and the relation (nd_3, nd_2) while loading nodes of type nd_3.

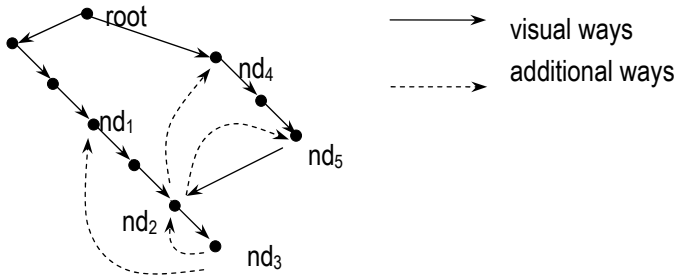

Figure 4. Paths assignment in the entity tree

In such a way, a tree is a kind of visual presentation of entities' relations and it reflects the view of the end user on the interconnection of the objects of data domain (or vice versa on the absence of the connection between some entities).

At last we get extended graph of the presentation model G'_{pr} (V'_{pr} – the set of its' nodes, E'_{pr} – the set of arcs), consisting of nodes and arcs of graphs G_{pr} and G_T; besides, this graph includes the set of arcs between object tree nodes and nodes-forms:

$$G'_{pr} = (V'_{pr}, E'_{pr}), \text{where } V'_{pr} = V_{pr} \cup V_T, E'_{pr} = E_{pr} \cup E_T \cup E_{ext}.$$

Conclusion

This work describes the base model of user interface of application, including objects of user interface and specifying interconnection between them. Based on described operations with graph of the presentation model there have been worked out additional algorithms that are not described in the work.

In order to realize suggested above graph model of user interface we can offer the approach of CASE-technology METAS, including in itself tools for automation of working out large scale IS, based on using multilevel structure of dynamically changing metadata [Ryzhkov, 2002]. Case-tools contain the set of program components, processing metadata of different levels and building on this base application, which has the meanings of interface designing and graphic interface of the end user of IS. Program components, realizing Windows-interface of applications [Kudelko, 2004], are built based on the present graph and they use algorithms suggested above.

Bibliography

[Kudelko, 2004] E.Y. Kudelko. The Application of Metadata for Automation of the User Interface Construction. In: Articles Collection of International Seminar "Modern Problems of Mechanics and Applied Mathematics", Voronezh University, Russia, 2004, pp. 310-313.

[Lyadova, 2003] L.N. Lyadova, S.A. Ryzhkov. CASE-technology METAS. In: Scientific Articles Collection "Mathematics of Program Systems". Perm University, Russia, 2003, pp. 4-18.

[Ryzhkov, 2002] S.A. Ryzhkov. The Concept of the Metadata in the Development of Information Systems. In: Scientific Articles Collection "Mathematics of Program Systems". Perm University, Russia, 2002, pp. 36-44.

Author's Information

Elena Kudelko – Perm State University, Department of Computer Science, Assistant; 15, Bukirev St., Perm, 614990, Russia; e-mail: kudelko_elena@mail.ru